

T S8/FULL/6

8/9/6 (Item 6 from file: 275)

DIALOG(R)File 275:Gale Group Computer DB(TM)

(c) 2005 The Gale Group. All rts. reserv.

01355132 SUPPLIER NUMBER: 08330674 (THIS IS THE FULL TEXT)

Retrospective on DACNOS. (prototype Distributed Academic Computing Network Operating System)

Geihs, Kurt; Hollberg, Ulf

Communications of the ACM, v33, n4, p439(10)

April, 1990

ISSN: 0001-0782 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABS

WORD COUNT: 9077 LINE COUNT: 00744

ABSTRACT: The Distributed Academic Computing Network Operating System (DACNOS), a prototype network operating system designed to facilitate distributed computing in heterogeneous environments, is described. DACNOS initially targeted the computer science department in West Germany's University of Karlsruhe, which includes IBM/370 mainframes, DEC VAX minicomputers and IBM PC microcomputers. These systems offered some file transfer but no resource sharing between applications running on different machines. DACNOS does not replace individual operating systems or reduce the autonomy of systems involved. It uses a Remote Service Call (RSC) programming interface and a 'remote like local' design principle. DACNOS' kernel provides communication and synchronization for low-level application to application cooperation. RSC provides a functionally complete interface for 'enabling' applications.

TEXT:

Retrospective on DACNOS In DACNOS (Distributed Academic Computing Network Operating System) we have addressed two characteristic aspects of today's computing environments: distribution and heterogeneity. While the former aspect is willingly accepted as a move to more power and flexibility for the user, the latter is in many environments a--sometimes unwanted, mostly functionally required--fact of life. Many people have studied distributed systems for the special case of a homogeneous environment. DACNOS has aimed at providing efficient and convenient support for the cooperation of heterogeneous computing systems.

Heterogeneity in systems is primarily because there is no single hardware and software architecture that serves all computing purposes equally well. Heterogeneity is apparent in different machine hardware architectures, operating systems, networking facilities, and user access

control, to name those who were considered for the DACNOS design. Other types of heterogeneity are conceivable, (e.g., user interfaces, application subsystems, or even multiple information media). These were not considered for the current prototype; however, they seem to make interesting areas for further research.

Our initial target environment was the computing infrastructure of the Computer Science department of the University of Karlsruhe in West Germany. [1] There we found, among others, IBM/370 computers running VM/CMS, DEC/VAX computers under VMS, [2] and IBM PCs running PC-DOS. The operating systems and machine architectures differ greatly in their hardware, software, and interface concepts. There was file transfer between the host computers, and some PCs were linked to the IBM hosts supporting terminal emulation, but there was no resource sharing between applications running on different machines. Data generated on one machine had to be shipped in a separate step to another machine in order to process it there in an application--not to mention services such as directories or transparent file access across the distributed computers. (Even today, this situation still is very typical for many data processing environments of large organizations.)

The DACNOS prototype was first implemented on VM/CMS, VMS, and PC DOS. This prototype has been studied extensively and used to build various distributed applications. It has also been ported to two more operating systems, i.e., AIX on IBM PC RT and IBM PS/2 and OS/2 on IBM PS/2. In this article we discuss the fundamental design assumptions of DACNOS that reflect our systematic approach to solve the heterogeneity problem as well as our experiences with implementing a prototype on top of five operating systems. For obvious reasons we cannot elaborate on all aspects and components of DACNOS in appropriate detail. The cited references should provide more information on a particular DACNOS subject. The section on "Goals and Implications" describes our design goals and constraints. The section on "Architecture" shows how the DACNOS architecture reflects these goals. The section on "Application Experiences" contains some examples for real-life applications that were built on top of DACNOS. In the section on "Implementation" we discuss what it takes to implement DACNOS on a system and how easy or hard the portation was for the above-mentioned operating systems. We also provide performance data for some scenarios. In the section on "Discussion of Related Work" we compare DACNOS to related work on heterogenous distributed systems. The last section contains our main conclusions and looks toward future extensions.

GOALS AND IMPLICATIONS

Resource sharing between heterogeneous autonomous computers has been the focus of our research. Fast networking hardware and low-level communication software was available to the academic community on the

campus, but it was still a very cumbersome and often replicated task to write an application that integrated services from several computing devices, although the need for such applications seemed to increase steadily. Our intention was to provide the programmer of a distributed application with convenient system support to facilitate the controlled access and distributed computing resources--basically the same support he or she is used to when writing a local application. This local-system paradigm was our guideline for many design decisions.

Two as-much-as-possible goals stood at the beginning of our design considerations:

(1) the application programmer should not have to deal with low-level details of operating systems and communication protocols and

(2) heterogeneity should be handled by the network operating system and be hidden from the application programmer.

In other words, we wanted to provide a high degree of insulation from both distribution and heterogeneity. These two goals led to the design of an application platform for programmers of distributed applications. This platform is called "Remote Service Call (RSC)." It is the key component in the NOS kernel that provides an application-oriented (as opposed to a communication-oriented) interface to the cooperation of heterogeneous computers. How the "application orientation" is reflected in the RSC interface will be shown in the next section.

Since the interconnected computing systems are part of many rather independent organizational university structures having individual application requirements and solution, (e.g., research institutes, library services, and student workstation pools), the following two design constraints were significant for the NOS design:

- * do not replace the individual operating systems and
- * retain the autonomy of the involved systems.

We could not and did not want to (and many commercial users would agree) enforce a single operating system, e.g., UNIX, [3] on all machines. (Several UNIX-based distributed systems can handle heterogeneous hardware architectures [3, 17, 20].) Such a step would have made a large base of applications and investments worthless. Consequently, our NOS is an add-on to the different operating systems that does not interface with existing applications, but makes it feasible to have access to remote resources in formerly only-local applications. In many cases this remote access is transparent to the application software. (See the section on "Application Experiences.")

When cooperating with remote partners, a DACNOS node does not give up its right to decide autonomously about the access and the management of its resources. For example, access to a resource has to be granted explicitly; it can be revoked at any time; and the allocation of resources to

requestors is completely up to the provider of the service. This property distinguishes the DAC Network Operating System from many distributed operating systems where the nodes relinquish some of their control autonomy to become part of a global "whole." The emphasis on autonomy does not preclude a DACNOS-wide management support that helps to allocate and control available resources in a desired manner [7]. It requires mechanisms for access control and protection across distributed computers.

When discussing software add-ons to heterogeneous systems: portability must be a design goal--not just for applications but also for the NOS software. Besides being good software engineering practice, portability is essential for a system that is to be ported to many heterogeneous computers. In our opinion we could otherwise not claim to have a systematic approach to overcome heterogeneity. To achieve portability one has to define a software module structure that clearly isolates system dependent and independent components. In so doing, the portation effort is reduced to the modification and adaptation of only a few components. A good modular structure will obviously also support the manageability and extensibility of such a rather large software complex.

ARCHITECTURE

Cooperation in DACNOS is conceived as client-server interactions. Clients and servers reside in logical nodes, which are mapped onto the (physically) nodes of the underlying communication network. A logical node has a network transport address and is typically associated with an operating system process or process group having a single address space. In VCM/CMS each virtual machine would be a logical node, while a single-user PC is considered a single logical node. In general, a logical node corresponds to a user process on a computer. It is the smallest addressable unit in the transport system.

Figure 1 shows the structure of a system with a DACNOS extension. Applications make use of the host operating system services as before and can now access remote resources through calls to the NOS System Services. Applications may also directly call operations offered by the NOS kernel's interface, i.e., the Remote Service Call interface. This is the lowest NOS access level. The components drawn below RSC in Figure 1 are not visible to the "outside."

Remote Service Call

RSC is the platform for distributed cooperation. It is coherently accessible on all logical nodes. The RSC interface is based on a set of objects that represent typical operating system primitives, i.e., requests, ports, storage, and accounts. RSC itself does not provide higher level objects like files or a directory, but provides the "building blocks" for making such objects accessible and manageable in the heterogeneous distributed environment. The RSC programming interface is a set of

operations, i.e., high-level language library functions, defined for the RSC objects. All operations are performed by a RSC worker that resides in the logical node and manages the RSC objects of this node as well as all communication required to access objects on another node.

Object sharing is the paradigm for distributed cooperation using RSC objects. To share an object (local or remote), access rights to the object are passed to the recipient who may then use the object in a way that is completely location, naming, and presentation transparent, i.e., just like a shared resource in a local system. Again, the local-system paradigm is the guideline behind the choice of RSC primitives and the object sharing. The system, i.e., the interacting RSC entities of DACNOS, provides the illusion of a single shared global object space.

As an example, consider a typical client-server interaction. The server program creates a port for its service and offers this port to certain clients. (Create and offer are RSC operations defined for the port objects.) The port is typed in the same sense possible requests and request data formats are specified by the server programmer and attached to the port. It is thus the handle for an abstract service object. The actual data conversion is performed transparently by the presentation component when the data is sent across the network.

In order to bind to the service represented by a port a client will have to explicitly issue a RSC share operation. To send a service request to the server the client creates an object called carrier that specifies the request and also contains value and reference parameters. Reference parameters in RSC are passed as access rights to RSC objects. For example, access to a data buffer at the client side could be granted with the Carrier for the duration of the request. (Data buffers are described by windows.) The DACNOS data presentation syntax notation comprises mechanisms to specify reference parameters as part of an interface description [6]. The type of user data contained in RSC objects, e.g., carrier and window, is defined by an attached type description string. This is used by RSC to perform the necessary conversions. The programmer specifies the data types in a language that is an extension of ASN.1 [10]. This notation is compiled into a more efficient internal description string.

It is important to note here that the programmer of an application will only have to think in terms of application-related operations, e.g., create port, share port, call service, read data, while RSC transparently performs all the required communication, error handling, access checks, data segmentation, data conversion, and even account management [7]. Shared objects make distributed programming "look and feel" like local programming. More information on RSC including a detailed description of its objects and implementation can be found in [6].

Global Transport and Kernel Service Call

Global Transport (GT) and Kernel Service Call (KSC) are those components that serve the portability of RSC. The portation of RSC is basically the portation of GT and KSC to the target machine.

GT is the coherent interface for the data transport need of RSC. The network is conceived as a set of interconnected islands of homogeneity with proprietary internal transport mechanisms. We did not impose a less efficient heavy-weight protocol where a specialized protocol performs much better. Thus, we use standardized protocols (OSI/TP-4, TCP/IP) only when a communication path crosses island boundaries.

GT basically is a reliable datagram interface to data transport between DACNOS logical nodes with a global OSI-style addressing scheme. RSC sees a simple "send-receive" interface where, for example, sequencing and duplication control are not required separately from the general error handling mechanisms in RSC. Internally, GT might very well use--depending on the transport protocol and network--connection-oriented services to send the datagrams between the network nodes. RSC, however, does not see connections on the transport level.

While GT handles diverse communication environments, KSC covers diverse host operating systems. Layered communication software typically deals with independent asynchronous events and, therefore, requires appropriate operating system support. With KSC we created an operating-system independent interface that offers communication-software-oriented services such as multiple light-weight processes, communication within shared memory, synchronization primitives for disjunctive multiple event handling, and timer services. Popular operating systems differ significantly in how much of this support is available and how it is offered to the applications. Consequently, the implementation efforts for KSC have ranged from "simple mapping of services" to "implementation of a coexistent multitasking system." (See the section on "Implementation.")

It is interesting to note that KSC is based on the same cooperation philosophy as RSC: KSC process cooperate through objects in shared local memory, and RSC processes cooperate through objects in a (virtually) shared global object space.

System Management Services

The DACNOS kernel provides communication, synchronization, and low-level access management support for application-to-application cooperation across a heterogeneous network. The DACNOS system management services complement these facilities with user-and resource-level services that help to organize and control the user access and resource allocation. These services are analogous to services in host operating systems and have to reside on nodes that are trustworthy.

The Directory Service maintains, distributes, and protects

information about network resources and services. It also contains a name service that controls the naming of resources in the network. The DACNOS directory is not specifically tailored for a specific application scenario, e.g., for message handling or file stores, but aims to be a universal directory capable of supporting a wide range of applications. An entry in the directory database is simply a tuple consisting of object name, entry owner, and a set of type specific attributes. Access control is performed on a per attribute basis by owner-controlled access lists [15]. The name space is divided into domains of unique names. A server would register its service at the directory specifying some name string. The name service part of the directory ensures the uniqueness of the name within the server's domain. Clients find out about a service by asking the directory for a particular service name. The directory's response contains the network address of a server that offers the desired service. Using the service name and network address the client establishes a binder to share the server port (share operation). After the share has succeeded the client will in subsequent operations only use the port handle that is a result parameter of the Share Port operation [15].

The Authentication and Authorization Service (AAS) provides for mutual identification of interacting partners and is a means for servers to control access to their services. It is the key component needed to cope with the problem of non-secure, freely accessible workstations in the network. The authentication is based on a password scheme, and it is assumed that a user cannot forge his or her transport address. Users have to log-in with their AAS before using DACNOS services. Thus a "DACNOS user-id" is then correlated with a transport address. To check the authorization of a client, servers present to the AAS the network address and user-id as given in the client request in order to decide about the validity of the purported identity [14].

The DACNOS Account Service is analogous to the accounting facilities of a local system. Resource consumption data is collected within the RSC kernel and shipped to the Account Server. RSC has a special account object to support these functions. The data provides valuable information on the utilization of network services and can be used in billing for a provided service. For example, there are many ways to limit the consumption of resources by certain users to user groups. A broad discussion of the problems of accounting and billing in heterogeneous environments and the DACNOS Account Server can be found in [7].

All DACNOS system services described here are distributed in a sense that more than one server may be involved in order to serve a service request. The reader should refer to the cited references to get more information on server interaction and protocols. On top of the management services there are applications that are also considered DACNOS system

services. These services, e.g., Remote File Access and Remote Execution, provides access to shared resources. They are discussed in the following section.

In a distributed system, more and new types of failures are possible than in the local case. DACNOS tries to assist the application programmer--as far as possible--in the handling of unexpected events. The representation of an invocation as a carrier make the relations between remote cooperating components explicit to the RSC kernel. A watchdog process inside the RSC kernel periodically checks the availability of the remote partners by sending probe messages. Loss of connection to a node and loss of a service will thus be detected. Failures are reported to the waiters on a carrier or a port by terminating their wait-state with a special return code. It is then the responsibility of the application to react reasonably within its context. Mechanisms for the coordination of distributed transactions are known and could be integrated into RSC. Nevertheless, the applications would have to be written to support a transactional behavior.

APPLICATION EXPERIENCES

Two operating-system related services were developed partly in parallel with the DACNOS kernel: Remote File Access and Remote Execution. Both provided feedback on the design of RSC and its interface. This feedback helped to improve its functionality and interface style.

Remote File Access

Remote File Access (RFA) is a global, homogeneous file system for heterogenous networks that provides transparent access to remote files [8]. According to our autonomy and transparency objectives, we do not replace any local file system, but accommodate the global RFA file system in the diverse local file systems. The RFA file system is partitioned into multiple RFA file servers, each being responsible for a subset of RFA files and RFA clients that mediate the user access to RFA. The running prototype supports sequential record-oriented files.

RFA servers use the local file systems of the host operating systems. This techniques minimizes the effort needed to port RFA servers to different operating systems. It also allows existing local files to be made available globally without copying ("publish"), thus allowing easy exchange of files between RFA and the local file system. Published local files should not be modified without RFA: otherwise their global consistency can not be guaranteed. The RFA client software is an extension and in some cases a modification of the local operating system. The extension offers access to the global RFA files through procedure and command interfaces. The modification opens the local file system interfaces for the global RFA files. It intercepts calls to the file system and re-routes them if necessary. Sets of global files can be bound ("mounted") inot the local

file system as "virtual volumes." Global files can be accessed transparently via their local aliases in the same ways as local files, and existing application programs may use global files without any change in the application code. This again is in line with our local-system paradigm.

The file naming structure of the global file system is hierarchical. Publication of a file at a file server includes assigning a global name to the published file. Similarly, binding a file into a local file system involves naming the file according to the local naming structure. Name mapping is assisted by user selectable default rules that can cover the most frequently used translations.

RFA uses RSC for cooperation between distributed RFA clients and servers. For example, an RFA file server offers a main port for "Open File" and maintenance-related requests. A private port together with a private server process is then created for each opened file. This port is shared with the client who uses it to access the file contents. These two reports differ in their visibility. The file server port is accessible to the public according to the defined access rights, yet an open file is a private matter between the client and the server. File data is exchanged between client and server using a shared RSC window object with the appropriate data description attached. Client and server only "read" or "write" to the arbitrarily sized window, while RSC handles the segmentation access control and the communication and conversion.

The mapping of open files to a port and a separate process has several advantages. The file server need not be concerned with dispatching. If multiple requests queue up, RSC contains the mechanisms to ensure a fair distribution of the file service among the clients. Authorization checking takes place during open time. For subsequent calls to the private port, the server can rely on the authorization of the caller. It is easy to add file service specific accounting to a server. Since the service providing process runs under the account of the client, this process can be charged for any desired account units. RSC and the accounting server collect the bills for the client process.

As to the implementation of RFA it is obvious that portability can only be limited since RFA client and server software anchor deep in the different host file systems--though they do not hinder the local functionality. Nevertheless, many RFA modules are portable based on the common programming language C and the RSC support. RFA clients with transparent remote file access are available for VM/CMS, VMS, and PC DOS. Designs for AIX and OS/2 have been done, but not yet implemented.

As mentioned before, RFA had influence on the RSC design. For example, a window's data format description originally was statistically defined where the window was created. This was insufficient in cases where the client creates a data window in his virtual memory without knowing the

actual record structure of the window that the server would use to write the file data. The server did not have means to specify data formats for the retrieved data whose structure was not known a priori to the client. This was changed in a way that one side can create a window with a wild-card data format description, which allows a sharer of the window to provide the structure information.

RSC has matured to become a powerful and convenient base for complex applications like RFA. Some of the advantages of using RSC for RFA are as follows. There is no need to design protocol elements for authorization, accounting, node failures, time out, or any other aspect of remote communication. This makes the interface design much easier and increases its stability. The data presentation functions of rSC are flexible enough to handle headers or trailers of variable length records transparent to RFA, i.e., without the need to reformat or mark retrieved data. Furthermore, the RSC window object is not only convenient, but also allows for transparent data transfer optimizations, e.g., a bulk transfer protocol is applied whenever it seems to be appropriate. This is transparent to the application that uses only Read window and Write window operations.

Remote Execution

The DACNOS Remote Execution Service (RES) [19] enables the sharing of programs located on remote computers. The aim was to build a natural and rather transparent extension to the invocation of local programs. Total transparency for remote execution is almost impossible to achieve in such an inherently heterogeneous environment, and we did not try to push the transparency limits. Rather, we wanted to provide a remote execution service that looks very familiar to the user.

Commonly, programs are designed by names. The user interface for starting a program has been extended such that the user can optionally append a server name to the program name where the program is to be executed. If no location is given with a program name the RES client part inquires at the directory service where this program is offered. The directories contain information about which programs are available on which computers. The user can specify a directory search scope in order to limit the search to certain nodes or domains. If a potential location was found, the execution request is sent to the given RES server.

Executing programs will require additional input and will output results. File access of a remote program is handled by the DACNOS RFA component. Thus, the remote program runs with the current working file directory of the user that requested the execution. All terminal I/O is intercepted and forwarded to the client and server side, respectively.

RES places only simple requirements on the data presentation since terminal I/O contains characters only, and other data access is handled by RFA. During the developments of RES, it was evident that the design and

implementation of interactions between clients and servers based on the RSC platform (plus RFA was almost trivial compared to the mastering of interceptions for terminal I/O and commands. These interception routines are obviously system dependent and not portable. Most of the rest of RES is portable. It has been implemented for VM/CMS, VMS, and PC-DOS (client only).

Database and Computation Server

To learn about the "ergonomics" of RSC and for demonstration purposes, remote access to an SQL database [23h and a high-precision numerical subroutine library were developed by summer students. In both cases, most of the design and all of the implementation was done by the students who had some programming experience but no DACNOS knowledge.

Within a few weeks after they had gained sufficient knowledge about the sub-systems to be accessed, they completed the implementation of a framework for the remote access using RSC objects with character data only and simulated access to the database or library. Without RSC this certainly would have required much more time in order to learn about the various interfaces, to develop the application and to debug it. Adding support for other data types took a matter of days. In both cases the client components were ported to all DACNOS systems. This was no effort at all, since the clients did not call any machine specific functions (only common C functions); and distributed cooperation was based on RSC.

IMPLEMENTATION

The design and implementation of DACNOS was a joint effort of researchers at the University of Karlsruhe and IBM. The project duration was limited to four years. All together it took an estimated 40 person-years to build the prototype as it is today.

All of our code has been written in the programming language C, except for some low-level KSC routines, which were better done in assembly language. Although the various implementations of C on heterogeneous machines have their compatibility problems, e.g., order of bit fields, alignment of structures and unions, default types, sign extension for shift operations, it was certainly the best available choice. It made our software highly portable, as long as certain conventions and rules were obeyed. A few numbers on the amount of code produced for the NOS kernel (excluding the DACNOS System Services) shall illustrate the development work: the KSC component (for VM/CMS) has about 7,000 lines of code, half C and half assembly language. The GT for VM/CMS consists of 6,500 lines of C code. Both figures vary depending on the host operating system. The NOS kernel, i.e., RSC including the data presentation, has roughly 55,000 lines of code and occupies 160K bytes of memory under VM/CMS. Host system dependencies of the RSC code are all completely separated into a collection of files, which have to be adapted when porting RSC.

KSC Implementation

As outlined in the section on "Architecture" KSC provides a coherent, communication-software-oriented surface on top of the host operating systems. It offers, among other facilities, light-weight processes sharing an address space. The portation of RSC is basically the portation/implementation of KSC. The amount of work varies depending on what is available in the host operating system.

For DACNOS on VM/CMS, each virtual machine (VM) is a "logical node" and represents an independent RSC entity with several internal and potentially many user-defined processes. We therefore had to add a transparent, coexistent, light-weight multitasking system to a VM, which originally did not offer support for multiple processes. It is important to note that KSC must not interfere with existing applications. Before KSC is added, the "CMS user process" is the only active thread in the VM. With KSC this view of the machine is still supported, but it is possible to create additional threads and thus multiplex the VM.

For VAX/VMS a logical node corresponds to a VMS process. With KSC this process can be split up into light-weight processes that share its address space. This implementation is analogous to the VM/CMS version, i.e., a VM in VM/CMS corresponds to a VMS process. The AIX version of KSC is also along this guideline [16] whereas OS/2 offers suitable facilities (light-weight multitasking with shared memory) that make the implementation of KSC basically a functional mapping.

PC DOS was important for us as a widely available low-cost system. A PC was considered a single logical node in DACNOS. KSC was implemented by mapping the KSC process constructs onto a multitasking system that was internally available in IBM for the PC. Although this was relatively easy, the PC eventually gave us a hard time because of memory size restrictions and lack of memory protection. Though we succeeded to port DACNOS to PC DOS and implemented the transparent Remote File Access client in the PC file system, the applicability of DACNOS on a PC is rather limited due to the severe lack of memory. DACNOS plus the RFA client leave free less than 70K of the 640K main memory of a fully equipped PC. We did not investigate the use of extended memory under DOS, which might give some relief from the storage problem, but still does not cure the lack of protection. Since OS/2 and AIX became available and better exploits the increased power of workstations, we did not further invest into the PC DOS version of DACNOS.

Performance

Performance has always been a high priority design goal for DACNOS. (Highest priority was to find a systematical and general solution for heterogeneity in distributed systems.) Given the DACNOS constraints and network environment our design could not exploit some of the mechanisms and techniques that have frequently been used in other projects on distributed

operating systems. For example, the DACNOS network may consist of a variety of interconnected local area networks and point-to-point lines with very diverse performance and reliability characteristics. The "cost" of a message is quite high. Therefore, extensive use of multicast or broadcast was impossible, and we tried to minimize the number of protocol messages without sacrificing functionality. Another example is the KSC, which is added on top of the host operating system. This obviously limits the KSC performance to the performance of the underlying general purpose system and is hardly comparable to an approach that builds a kernel directly on the bare hardware. These approaches make different assumptions and aim at different goals than DACNOS.

We measured the performance of selected RSC interactions involving various machines and communication links. The host measurements were taken during regular daytime use with light to medium load on the hosts. Some examples:

- * The measured round trip time of an empty request between two VMs on an IBM 4361 was 50 milliseconds (msec). About the same time was observed on a VAX 8600.

- * The same request between two VMs on two separate IBM/370 machines (4361 and 3083) connected by a 64K bits-per-second link using a proprietary protocol took approximately 125 msec, yet it took 210 msec on two VAXs (8600 and 8300) with VMS connected by an Ethernet and DECnet protocols.

- * For two PC/AT personal computers on a token ring the empty request took 165 msec to complete, and PC to IBM 4361 host via token ring and gateway took 345 msec.

When more user data is sent with the request the execution time is the sum of the above-fixed amount for the empty request plus an increase proportional to the speed of the communication link.

We were also very much interested in the overhead introduced by the DACNOS kernel compared to the basic, unenhanced inter-process communication facilities of the host system. This is best expressed in the approximate number of machine instructions. For a round trip request the RSC client performs 6,000, the server side 8,000 instructions. GT (including KSC, but excluding the transport protocol itself) adds another 1,500 instructions for a send operation and 6,000 for receive. For the above-mentioned scenario with the client on an IBM 4361 (1.5 million instructions per second (Mips)) and a server on an IBM 3083 (5 Mips), this amounts to roughly 12 milliseconds for RSC and GT, a number we were quite satisfied with. If both client and server are located on the 4361 and only VM/CMS internal communication is used, the processing for RSC and GT takes roughly 20 msec and process switches, interrupt handling, and data copy operations, take the rest, i.e., 30 msec.

DISCUSSION OF RELATED WORK

In this section we restrict our discussion to related work that concentrates like DACNOS on adding solutions for heterogeneity and distribution to existing computing environments. We do not discuss distributed operating systems like March [1] or Locus [20], which can also work on heterogeneous hardware, but are built on the bare hardware. They are not meant to coexist with a local host operating system. Here we will discuss systems that are extensions to existing host operating systems.

Cooperation in heterogeneous distributed environments is facilitated by introducing a unified view onto the heterogeneity that is an abstraction from the given dissimilarities. Several locations for such an abstraction layer are conceivable. For the application programmer using DACNOS this abstraction is given at the RSC interface. (Other internal abstraction levels are the KSC and GT interfaces. These facilitate the portation of DACNOS, but are not seen by the "RSC programmer.") The RSC cooperation facilities, i.e., the RSC objects and operations, are coherently supported by all DACNOS nodes and were designed analogous to the facilities of a local operating system.

Another approach is to move the unification layer into certain "key" applications. In HCS [18] "key facilities," i.e., remote procedure call (RPC), naming and binding, are made compatible (or newly created, if not available) across the various types of systems. On top of these facilities common services are implemented that are considered most important: file store, mail, printing, and remote computation. In HCS, heterogeneity is "accommodated" into certain services, but it is "abstracted" in DACNOS at the operating system level.

A project that shares the emphasis on heterogeneity and the need for a comprehensive solution with DACNOS is Cronus [21]. Object orientation, the integration of access protection into the kernel, a network independent transport interface, software portability, and operating system add-on rather than replacement are common properties for both systems. Though the design objectives are very similar, there are a number of differences in the approach and the implementation. For example, the Cronus object is on another level than the basic "building block" objects of RSC. There are object managers for each type of Cronus object while there is only one RSC entity that manages the objects of an RSC node. Cronus objects can be replicated and can migrate, which requires extensive use of group- and multicast search operations. Therefore, it is practically essential to have adequate communication support, i.e., a fast LAN with broadcast capabilities. RSC objects cannot be moved or replicated. The RSC protocol does not use broadcast or multicast facilities, although the DACNOS Directory and Orientation Service occasionally will have to issue search operations. Cooperation under RSC is based on the object sharing paradigm: the programmer thinks in terms of sharing objects just as in the local

case, and message passing is the implicit mechanism used to implement such an "illusion" in the distributed environment. In Cronus, cooperation is achieved using explicit message-passing primitives and the interaction style is much more communication-oriented.

Sun RPC [24] and Apollo NCS [2] are representatives of Unix-based RPC packages. Both use the services of the underlying host operating system and transport service and support heterogeneity. In NCS, client and server stubs are generated automatically by a compiler from interface descriptions written in a "Network Interface Description Language." The Sun RPC only provides an extensible set of library routines for the marshalling and demarshalling of parameters. It is the responsibility of the application programmer to make the appropriate calls. In DACNOS, the invocation of remote operations is done by sending an RSC carrier. RSC, however, supports a different, more flexible and powerful interaction model. Requests are associated with carrier objects that may contain data values and object references as parameters (see the section on "Architecture"). Sending a carrier is an asynchronous operation. Therefore, clients may have several carriers outstanding and may selectively wait for their completion. Servers may receive and work on multiple requests simultaneously. The carrier also contains automatically appended information for system management purposes, e.g., authorization, accounting, and dispatching priorities [7]. In all three systems various services are built on top of these communication kernels. Their differences are not discussed here, because the major focus of this article is on the kernel of DACNOS.

There are also language-based approaches to conquer heterogeneity. In DAPHNE [13] coherence is achieved at the programming language level. Components of a program can easily be distributed for execution on different nodes of a heterogeneous network. The means for cooperation between the distributed components is a Remote Procedure Call (RPC) that is adapted to handle the heterogeneity. It is supported by a stub generator and appropriately modified run-time environment. In [5] a programming language (Network Command Language (NCL)) was defined for the description of remote service requests. Using pre-defined function libraries and additional server specific functions NCL expressions are created and shipped to a server. (A canonical data representation solves the data incompatibility problem.) With command language expressions, a client can "program" the server to perform a sequence of functions in a single request avoiding the overhead of multiple remote procedure calls.

We would also like to mention two other very prominent attempts at mastering heterogeneity: Open Systems Interconnection (OSI) and IBM's Systems Application Architecture (SAA). OSI defines standards for the communication between heterogeneous computer systems [9]. The set of standards is still evolving. Only lately, efforts have been started to

define a platform for distributed applications that goes beyond the mere communication aspects of distributed processing [11]. DACNOS is considered one of the prototypes for a support environment for ODP, which is being developed by the European Computer Manufacturers Association (ECMA) [4]. The ODP activities will produce a reference model about how to integrate and describe the various aspects of distributed computing like communication, directories, security and management. Such a framework, however, will not provide specifications for the implementation on a real system or portability considerations, i.e., problems that were solved in DACNOS and related research projects.

SAA is a software architecture for the development of consistent applications across the major IBM computing architectures [25]. SAA specifies common interfaces and conventions for user access, communication, and programming on dissimilar operating systems. Benefits of such an architecture will be easy migration between systems, the portability of software, and the elimination of redundant development efforts. DACNOS has tackled the subset of the SAA objectives, which are related to distributed processing in a slightly different, historically grown, mixed vendor computing environment. Nevertheless, the DACNOS prototype demonstrates not only the feasibility but also the potential benefits of SAA [22].

CONCLUSIONS AND OUTLOOK

The motivation for DACNOS stemmed from the demand for resource sharing in historically grown heterogeneous computer networks and the fundamental lack of appropriate support for such applications. Our main goal was to provide system level support that takes most of the burden of distribution and heterogeneity away from the programmer of a distributed application. The DACNOS prototype demonstrates the feasibility of such an approach.

Feedback from applications on DACNOS has provided us with valuable insights into our design decisions. The more general observations are: First, we have found that the "remote-like local" design principle makes the interfaces easy to comprehend for application programmers. This familiarity speeds up the development process and increases the productivity. Second, the integration of access and access management into the kernel relieves the programmer from explicitly dealing with much of the (in one way or another) required access management. Separating management functions and shifting them into the NOS kernel avoids costly "reinventions of the wheel." Third, the RSC object interface has proven to be a functionally complete "application-enabling" interface. This style of interface is not necessarily tied to the current implementation of DACNOS.

We could well imagine having such an enabler for distributed applications on top of other data transport environments, possibly implemented on a kind of host operating system support other than KSC. We

claim that transport level primitives are too low-level for application programmers while the remote procedure call unnecessarily imposes a certain programming style that is often inadequate and sometimes cumbersome for the cooperation of independent networked processors. RSC is located somewhere in between the two offering an application-oriented abstraction that integrates communication as well as management functions. Finally, on a platformlike DACNOS application programs are potentially portable whether or not their scope is only local or remote. This clearly goes beyond the mere ability to communicate with other heterogeneous systems by using standardized communication protocols. DACNOS has most of the functionality that is being discussed in OSI standardization activities; plus, it presents a solution for system integration and software engineering problems.

Such a comprehensive support is not free. The complexity of the NOS kernel's design and implementation is higher than for approaches that run under the "keep it simple" mode. Less functionality in the kernel, however, tends to lead to replication of development efforts, less coherency between components, and thus, potentially reduced interoperability. Enforcing mechanisms in the kernel is the canonical approach with a higher complexity for the system designer whereas the noncanonical approach shifts some of the complexity to the application programmer with all the potential pitfalls--and some potential performance and optimization benefits. The DACNOS prototype, however, demonstrates that a functionally complete, more complex kernel can still perform very well.

The DACNOS development has taught us that in order to manage the complexity of a design effort it was extremely helpful to have a clear initial design guideline, i.e., in our case the local-system paradigm. This brought orientation in the early stages of the design process. It also brought consistency as we went along with the system design. And it helped to structure the initially huge problem space. No one does it completely right the first time. So it is almost needless to say that we would make several technical modifications and extensions if we had to do it again. None of these, however is related to our overall approach and design philosophy.

For example, we underestimated the problems of coordinating software development, versions, and maintenance on different computers by several people at different places. Usage of a software control system from the beginning of the project would have helped to eliminate several misunderstandings, incompatibilities and duplicated efforts caused by version-handling errors. We now think that CASE tools should be integrated into the development and implementation process. The implementation work of the DACNOS kernel was structured vertically, i.e., responsibilities were divided by the type of the computer system. Thus, the developers had to

know internals of all kernel components. A horizontal structure would assign responsibility by component. This requires the developer to know several systems, but system interfaces are stable compared to the internals of software components under development. Therefore, a horizontal structure is potentially more efficient. KSC was designed to isolate local operating system dependencies from the rest of the DACNOS software. Full screen input and output to the user's terminal were not included. We underestimated its impact on the structure and portability of reapplications. Today, we would consider the integration of an existing window-oriented interface into KSC.

The development of DACNOS is basically finished. The prototype is in use at various locations outside of the IBM ENC. It was selected as a development basis for distributed applications because of its unique functionality and flexibility. Among the external users are two European universities and an European RACE project. So far, user reaction is very positive.

Our experiences with usability and performance were confirmed. Currently, there are not enough users to publish statements on the scalability of our design. We intend to provide such data in a future report. There are still some activities going on to complete the portation of DACNOS System Services to the operating systems AIX and OS/2. We see opportunities for further extensions to DACNOS-like transactions, fault-tolerance, support for distributed debugging, and support for distributed applications that incorporate multiple information media like data, voice, and video. It is unclear to us what kind of system support suits the programming of these applications and whether DACNOS can be useful as a starting point for such a system support. These questions will be the focus of our future research.

Acknowledgments. Many people, too numerous to mention, have contributed to the DACNOS project. Special thanks go to Hermann Schmutz for design contributions and management guidance and to Herbert Eberle for his design and invaluable implementation contributions. The following persons have contributed significant parts of the code of the DACNOS prototype: P. Brecht, H. V. Drachenfels, C. Foerster, G. Harter, A. Kaemmer, B. Kieser, E. Kraemer, B. Mattes, S. Mengler, H. Moons, M. Philippsen, R. Oechsle, M. Salmony, B. Schoener, A. Schill, M. Seifert, P. Silberbusch, R. Staroste, M. Wasmund, G. Wild, H. Zoeller. We are grateful to all of them.

(*1) The DACNOS effort was part of a cooperation project of the University of Karlsruhe and IBM Germany. The project's goal was to study the application of computers for the support of academic teaching and research [12].

(*2) DEC, VAX and VMS are registered trademarks of Digital Equipment Corporation.

(*3) UNIX is a registered trademark of AT&T Bell Laboratories.

REFERENCES

- [1] Acetta, M., et al., Mach: A new Kernel Foundation for UNIX development. In Proceedings of Summer Usenix Conference (Atlanta, Ga., June 9-13). USENIX Assoc., Berkeley, Calif., pp. 93-112.
- [2] Apollo Computer Inc. Network Computing System Reference Manual. Chelmsford, Mass., 1987.
- [3] Balkovich, E., Lerman, S., and Parmelee, R. P. Computing in higher education: The Athena Experience. Commun. ACM 28, 11 (Nov. 1985), 1214-1224.
- [4] ECMA Support Environment for Open Distributed Processing (SE-ODP). ECMA Tech. Rep., No. 49, Geneva, Switzerland, 1990.
- [5] Falcone, J. R. A programmable interface language for heterogeneous distributed systems. ACM Trans. Comput. Syst. 5, 4 (Nov. 1987), 330-351.
- [6] Geihs, K., et al. An architecture for the cooperation of heterogeneous operating systems. In Proceedings of IEEE Computer Networking Symposium (Washington, D.C., Apr. 11-13). IEEE, New York, 1988, pp. 300-312.
- [7] Harter G., and Geish, K. An accounting service for heterogeneous distributed environments. In Proceedings of the Eighth International Conference on Distributed Computing Systems (San Jose, Calif., June 13-17). IEEE, Net York, 1988, pp. 207-214.
- [8] Hollberg, U., Schmutz, H., and Silberbusch, P. Remote File Access: A distributed file system for heterogeneous networks. In Proceedings of the GI/NTG Conference on Communication in Distributed Systems (Aachen, West Germany, Feb. 16-20). Springer Verlag, New York, 1987, pp. 293-310.
- [9] ISO OSI: Open Systems Interconnections Basic Reference Model. International Standard 7498, Geneva, Switzerland, 1984.
- [10] ISO Specification of Abstract Syntax Notation 1 (ASN.1). International Standard 8824, Geneva, Switzerland, 1987.
- [11] ISO ODP: Open Distributed Processing, Reference Model. In preparation at working group ISO/IEC-JTC1-SC21-WG7, Geneva, Switzerland.
- [12] Krueger G., and Mueller, G., (Eds.) HECTOR. Vol. I and II, Springer Verlag, New York, 1988.
- [13] Loehr, K. P., Mueller, J., and Nentwig, L. DAPHNE: Support for distributed applications programming in heterogeneous networks. In Proceedings of Eighth International Conference on Distributed Computing Systems (San Jose, Calif., June 13-17). IEEE, New York, 1988, pp. 63-71.
- [14] Mattes, B. Authentication in resource sharing networks. In HECTOR, G. Krueger and G. Mueller, Eds. Vol. II, Basic Projects. Springer Verlag, New York, 1988, pp. 126-139.
- [15] Mattes B., and Drachenfels, H. V. Directory and orientation in

heterogeneous networks. In HECTOR, G. Krueger and G. Mueller, Eds. Vol. II, Basic Projects. Springer Verlag, New York, 1988, pp. 110-125.

[16] Moons, H., Verbaeten, P., and Hollberg, U. Distributed computing in heterogeneous environments. In Proceedings of European Unix Users Group (EUUG) Spring '90 Conference (Munich, West Germany, April 23-27) 1990.

[17] Morris, J. H., et al. Andrew: A distributed personal computing environment. Commun. ACM 29, 3 (Mar. 1986), 184-201.

[18] Notkin, D., et al. Interconnecting heterogeneous computer systems. Commun. ACM 31, 3 (Mar. 1988), 258-273.

[19] Oechsle, R. A remote execution service in a heterogeneous network. In HECTOR, G. Krueger and G. Nueller, Eds. Vol. II, Basic Projects. Springer Verlag, New York, 1988, pp. 169-182.

[20] Popek, G. J., and Walker, B. J. (Eds.) The LOCUS Distributed System Architecture. MIT Press, Cambridge Mass., 1985.

[21] Schantz, R. E., Thomas, R. H., and Bono, G. The architecture of the Cronus Distributed Operating System. In Proceedings of the Sixth International Conference on Distributed Computing Systems (Cambridge, Mass., Feb. 22-24). IEEE, New York, 1986, pp. 250-259.

[22] Scheer, A. L. SAA Distributed Processing. IBM Syst. J. 27, 3 (1988), 370-383.

[23] Schoener, B., and Kieser, B. TRansparent database access in a network of heterogeneous systems. In Proceedings of the GI/NTG Conference on Communication in Distributed Systems (Stuttgart, West Germany, Feb. 22-24). Springer Verlag, New York, 1989, pp. 415-429.

[24] Sun Microsystems External Data Representation Reference Manual. Mountain View, Calif., 1985.

[25] Wheeler, E. F., and Ganek, A. G. Introduction to Systems Application Architecture. IBM Syst. J. 27, 3 (1988), 250-263.

KURT GEISH is currently a research staff member in the Operating Systems Research Group of the IBM European Networking Center, Heidelberg, Germany. His current research interest include heterogeneous distributed systems, open distributed processing (ODP), and application for highspeed networks.

ULF HOLLBERG is currently a research staff member in the Operating Systems Research Group of the IBM European Networking Center. His current research interests include heterogeneous distributed systems, distributed file systems, and application programming interfaces. Authors' Present Address: IBM European Networking Center, Tiergartenstrasse 8, D-6900 Heidelberg, Germany, GEIHS@HDIBM1.BITNET and HOLLBERG@DHDIBM1.BIT
COPYRIGHT 1990 Association for Computing Machinery Inc.

DESCRIPTORS: Multivendor Systems; Distributed Processing; Network Operating System; Prototype; LAN; Education

FILE SEGMENT: AI File 88

?

Refine Search

Your wildcard search against 10000 terms has yielded the results below.

Your result set for the last L# is incomplete.

The probable cause is use of unlimited truncation. Revise your search strategy to use limited truncation.

Search Results -

Terms	Documents
L3 and mess\$	3

Database:

US Pre-Grant Publication Full-Text Database
 US Patents Full-Text Database
 US OCR Full-Text Database
 EPO Abstracts Database
 JPO Abstracts Database
 Derwent World Patents Index
 IBM Technical Disclosure Bulletins

Search:

L4

Refine Search

Recall Text

Clear

Interrupt

Search History

DATE: Thursday, May 12, 2005 [Printable Copy](#) [Create Case](#)

<u>Set</u> <u>Name</u> side by side	<u>Query</u>	<u>Hit</u> <u>Count</u>	<u>Set</u> <u>Name</u> result set
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; THES=ASSIGNEE; PLUR=YES; OP=OR		
<u>L4</u>	L3 and mess\$	3	<u>L4</u>
<u>L3</u>	L2 and collat\$	11	<u>L3</u>
<u>L2</u>	L1 and @a\$<=20030110	166	<u>L2</u>
<u>L1</u>	map\$ same (collat\$ or correlat\$) same (zon\$ or area\$ or region\$) same address\$	224	<u>L1</u>

END OF SEARCH HISTORY

SHOW FILES

File 275:Gale Group Computer DB(TM) 1983-2005/May 12

(c) 2005 The Gale Group

File 634:San Jose Mercury Jun 1985-2005/May 11

(c) 2005 San Jose Mercury News

File 256:TechInfoSource 82-2005/Mar

(c) 2005 Info.Sources Inc

File 2:INSPEC 1969-2005/Apr W4

(c) 2005 Institution of Electrical Engineers

File 6:NTIS 1964-2005/May W1

(c) 2005 NTIS, Intl Cpyrght All Rights Res

File 8:Ei Compendex(R) 1970-2005/May W1

(c) 2005 Elsevier Eng. Info. Inc.

File 211:Gale Group Newsearch(TM) 2005/May 12

(c) 2005 The Gale Group

File 239:Mathsci 1940-2005/Jun

(c) 2005 American Mathematical Society

File 674:Computer News Fulltext 1989-2005/May W2

(c) 2005 IDG Communications

File 675:TRADEMARKSCAN(R)-Sweden 2005/May W2

(c) 2005 THOMSON COMPUMARK

?

Set	Items	Description
S1	2	((PICTURE? (9N) TEXT?) (9W) (CORRELAT? OR COMPAR? OR MA OR REPLAC? OR SUBSTITUT?)) AND DIAGNOS? AND PY<2005
?		

SHOW FILES

File 275:Gale Group Computer DB(TM) 1983-2005/May 12

(c) 2005 The Gale Group

File 634:San Jose Mercury Jun 1985-2005/May 11

(c) 2005 San Jose Mercury News

File 256:TecInfoSource 82-2005/Mar

(c) 2005 Info.Sources Inc

File 2:INSPEC 1969-2005/Apr W4

(c) 2005 Institution of Electrical Engineers

File 6:NTIS 1964-2005/May W1

(c) 2005 NTIS, Intl Cpyrght All Rights Res

File 8:Ei Compendex(R) 1970-2005/May W1

(c) 2005 Elsevier Eng. Info. Inc.

File 211:Gale Group Newsearch(TM) 2005/May 12

(c) 2005 The Gale Group

File 239:Mathsci 1940-2005/Jun

(c) 2005 American Mathematical Society

File 674:Computer News Fulltext 1989-2005/May W2

(c) 2005 IDG Communications

File 675:TRADEMARKSCAN(R)-Sweden 2005/May W2

(c) 2005 THOMSON COMPUMARK

File 80:TGG Aerospace/Def.Mkts(R) 1982-2005/May 12

(c) 2005 The Gale Group

File 63:Transport Res(TRIS) 1970-2005/Apr

(c) fmt only 2005 Dialog Corp.

?

Set	Items	Description
S1	0	MAP?AND (ADDRESS? (4N) (CORRELAT? OR COLLATE?)) AND T T? AND PD<=030110
S2	1	MAP? AND (ADDRESS? (4N) (CORRELAT? OR COLLATE?)) AND T ET? AND PD<=030110
S3	8	MAP? AND ((ZON? OR REGION? OR AREA?) (4N) (CORRELAT? OR LLATE?)) AND TARGET? AND PD<=030110
S4	8	RD (unique items)
S5	8	RD (unique items)
?		

T S6/3,KWIC/1-5

6/3,KWIC/1 (Item 1 from file: 275)

DIALOG(R)File 275:Gale Group Computer DB(TM)

(c) 2005 The Gale Group. All rts. reserv.

02609742 SUPPLIER NUMBER: 87012022 (USE FORMAT 7 OR 9 FOR FULL)

Theory of keyblock-based image retrieval.

Zhu, Lei; Rao, Aibing; Zhang, Aidong

ACM Transactions on Information Systems, 20, 2, 224(34)

April, 2002

ISSN: 1046-8188 LANGUAGE: English RECORD TYPE: Fulltext; Abstract

WORD COUNT: 12987 LINE COUNT: 01072

... compressed based on vector quantization (VQ) (Gersho and Gray 1992), then a code vector usage map or a code vector histogram is used as the image features. The work is a...to) i (less than or equal to) N , are the keyblocks. Let F be a mapping :

$F : (R.\text{sup}.k) \text{ (right arrow) } C = \{(c.\text{sub}.1), \dots, (c.\text{sub}.i), \dots, (c.\text{sub}.j) \text{ (member of) } (R.\text{sup}.k)\}$, the mapping F gives rise to a partition of T which consists of N cells $P = \{(p...$

...sub.j) and output code (c.sub.i) (for example, the Euclidean distance), an optimal mapping should satisfy the following conditions:

--Nearest Neighbor Condition: For each (p.sub.i), if t...that the above models focus only on the appearance of individual keyblocks in images. The correlations between keyblocks are not addressed .

4.2 Correlation -Enhanced Models

We will now discuss the integration of keyblock correlations in the feature models...of New York at Buffalo

This research is supported by NSF and National Imaging and Mapping Agency (NIMA). URL of demo: <http://vangogh.cse.buffalo.edu:8080/>.

Authors' address: Department of...

20020401

6/3,KWIC/2 (Item 2 from file: 275)

DIALOG(R)File 275:Gale Group Computer DB(TM)

(c) 2005 The Gale Group. All rts. reserv.

02517006 SUPPLIER NUMBER: 76156234 (USE FORMAT 7 OR 9 FOR FULL)

Quova's GeoPoint.(Company Business and Marketing)

10/718670

Set	Items	Description
S1	0	MAP?AND (ADDRESS? (4N) (CORRELAT? OR COLLATE?)) AND T T? AND PD<=030110
S2	1	MAP? AND (ADDRESS? (4N) (CORRELAT? OR COLLATE?)) AND T ET? AND PD<=030110
S3	8	MAP? AND ((ZON? OR REGION? OR AREA?) (4N) (CORRELAT? OR LLATE?)) AND TARGET? AND PD<=030110
S4	8	RD (unique items)
S5	8	RD (unique items)
S6	5	MAP? AND (ADDRESS? (6N) (CORRELAT? OR COLLATE?)) AND P 030110
S7	0	MAP?AND (ADDRESS? (4N) (CORRELAT? OR COLLATE?)) AND T T? AND PY<=2003
S8	6	MAP? AND (ADDRESS? (4N) (CORRELAT? OR COLLATE?)) AND T T? AND PY<2003
?		

T S2/3,KWIC/1

2/3,KWIC/1 (Item 1 from file: 275)

DIALOG(R)File 275:Gale Group Computer DB(TM)

(c) 2005 The Gale Group. All rts. reserv.

02517006 SUPPLIER NUMBER: 76156234 (USE FORMAT 7 OR 9 FOR FULL)
Quova's GeoPoint.(Company Business and Marketing)

Carr, Jim

Network Magazine, 32

July 1, 2001

ISSN: 1093-8001 LANGUAGE: English RECORD TYPE: Fulltext; Abstract

WORD COUNT: 936 LINE COUNT: 00079

...ABSTRACT: pinpoint the geographic location of visitors to their Web sites. It also enables them to target content and marketing messages by country, metropolitan area, Direct Marketing Area (DMA) and by zip...

...in the US. The service is based on a global network of 60 servers that maps IP addresses to geographic locations.

... to their sites.

The privately held Redwood City, CA company's GeoPoint permits users to target their content, marketing messages, and product mix by country, Direct Marketing Area (DMA), metropolitan area...

...the playing field between Web sites and radio and television stations and newspapers for delivering targeted , localized advertising and content of all kinds.

But the product offers more than just the...

...Media has incorporated GeoPoint into its AdVariant product, which permits advertisers to create and position targeted banner ads.

According to Pat Transue, the company's VP of engineering, the GeoPoint service...

...moving to GeoPoint, Amazing Media employed five IT staffers to operate and manage a homegrown mapping facility. Deploying GeoPoint allowed the company to move these workers into "more strategic" positions, says...

...SERVERS

Quova's GeoPoint service is based on a global network of 60 servers that maps IP addresses to physical locations, according to Alexander. In the "commercial" part of the Web...

...and government sites and IP addresses inside corporate networks, the company has collected, analyzed, and mapped the IP address and geographic location information on about 1.4 billion IP addresses, she...

...Web surfer hits a GeoPoint-enabled site, that API communicates with Quova's DDS, which correlates the IP address into a geographical location and returns the information to the customer server, which can then present a targeted ad, content, or whatever.

Depending on the "need for speed," the customer can manage the...

20010701

?

4/3,KWIC/1 (Item 1 from file: 275)

DIALOG(R)File 275:Gale Group Computer DB(TM)

(c) 2005 The Gale Group. All rts. reserv.

02499319 SUPPLIER NUMBER: 73889658 (USE FORMAT 7 OR 9 FOR FULL)
Digging for the Root Cause of Network Problems -- Intelligence engines are coming of age. How will they change the network management industry, as well as your software options?(Industry Trend or Event)

Drogseth, Dennis

Network Magazine, 96

May 1, 2001

ISSN: 1093-8001 LANGUAGE: English RECORD TYPE: Fulltext; Abstract

WORD COUNT: 2951 LINE COUNT: 00269

... alarm suppression. Some vendors are also using topological insight for inventory and asset management.

Another area of change involves advanced correlative intelligence. Root-cause analysis requires a way to correlate across events, topological change, and other...

...service-level solutions, and sometimes to business process implementation, where availability and performance failures are mapped to service and business impact. In this way, the power of the intelligence engine can...

...comprised of components called Early Warning Center, Report Center, and Switched Early Warning Center-is targeted at NOC and IT infrastructure engineering personnel. The product can address network brownouts through a

...

...autodiscovery and topological awareness.

CAP-TREND focuses on performance and capacity planning, and ADVANTAG is targeted at service-level management. CAP-TREND and COORDINATOR are easy to install and keep current...

...service-level and alarm control; Netcool/Precision, which performs root-cause analysis; and Netcool/Impact, targeted toward operationally focused tuning of service impact, including application services.

Micromuse products are beginning to...

...have historically been network-centric, focusing on service providers and large enterprises. They don't target out-of-the box deployment, but

fare well where they're deployed strategically.

RiverSoft Technologies...context can help solve such misrepresentations.

To accommodate highly distributed Internet environments, Tavve has also targeted firewall requirements with its ePROBE architecture. ePROBE works inside firewalls as a distributed capability and...

...reached at drogseth@enterprisemanagement.com.

Resources

There are precious few resources in the industry truly targeted at the root-cause-analysis space, except for research and vendor sites. Below are some...

20010501

4/3,KWIC/2 (Item 2 from file: 275)

DIALOG(R)File 275:Gale Group Computer DB(TM)

(c) 2005 The Gale Group. All rts. reserv.

02475288 SUPPLIER NUMBER: 68658853 (USE FORMAT 7 OR 9 FOR FULL
Telecommunications.(Directory)

Technology & Learning, 21, 5, 44

Dec, 2000

DOCUMENT TYPE: Directory ISSN: 1053-6728 LANGUAGE: English

RECORD TYPE: Fulltext; Abstract

WORD COUNT: 833 LINE COUNT: 00076

... companion to the video series A Biography of America. Although the Web site's 26 areas correlate specifically with the video series, they can also be used independently. Each section of the site includes a list of key events, a map, a transcript of the corresponding video program, a bibliography, and an annotated list of Web...

...much of his life, is complemented by introductions and annotations as well as historical background, maps, and illustrations. The diaries are available in the American Memory section of www.loc.gov...

...training source for mastering Macromedia products. Macromedia University offers affordable and convenient e-learning solutions targeted at developers of every skill level. Courses are self-paced and demonstrate timesaving tips and...

20001201

4/3,KWIC/3 (Item 1 from file: 674)

DIALOG(R)File 674:Computer News Fulltext
(c) 2005 IDG Communications. All rts. reserv.

078559

The Hacker in All of Us

In which our intrepid reporter learns how easy it is to suck the guts out of a victim's server -- and how much sheer, unrelenting fun By Deborah Radcliff

Journal: Computerworld Page Number: 78
Publication Date: October 11, 1999
Word Count: 1627 Line Count: 140

Text:

...says El Paso Energy's Norwood.

We deploy a few common network troubleshooting tools (like zone transfers -- normally used to correlate data between the backup and primary servers, and Name Service lookup -- a utility used to...

...on those ports.

I'm particularly taken with the stealthy Nmap, a utility for network mapping available for free off the Web. We deploy Nmap against our primary target to get a road map of open ports, along with the network protocols and application services they support.

At the...

...and Schultze, 31, proves he's worthy of his name.

We start by picking our target. Test servers are notorious for lax password controls and monitoring. Or we could sniff the...

4/3,KWIC/4 (Item 1 from file: 80)

DIALOG(R)File 80:TGG Aerospace/Def.Mkts(R)
(c) 2005 The Gale Group. All rts. reserv.

01093465 Supplier Number: 39790964

Tomahawk: The US Navy's New Option

Navy International, v91, n7, p394-398

July, 1986

Language: English Record Type: Abstract

Document Type: Magazine/Journal; Trade

ABSTRACT:

...attitude/heading reference system, and the missile also features a modified Harpoon missile seeker for target reacquisition and terminal guidance. The other 2 versions employ an INS for the over-water...

...TERCOM), combining a radar altimeter and a microprocessor to compare the land below to digitised maps stored in the system memory. The INS-TERCOM combination is designed to bring the weapon to within a reported 100 ft of the target, which is adequate for TLAM/N. The TLAM/C conventionally warhead requires greater accuracy, and thus employs a digital scene matching area correlator in the terminal flight phase.

...

19860701

4/3,KWIC/5 (Item 2 from file: 80)

DIALOG(R)File 80:TGG Aerospace/Def.Mkts(R)

(c) 2005 The Gale Group. All rts. reserv.

01046450 Supplier Number: 39444661

PERSHING GUIDANCE SYSTEM DELIVERS UNPRECEDENTED ACCURACY

Defense Electronics, v16, n12, p56

Dec, 1984

Language: English Record Type: Abstract

Document Type: Magazine/Journal; Trade

ABSTRACT:

...a Goodyear Aerospace terminal guidance system that increases missile accuracy, allowing it to destroy hardened targets with a single warhead smaller than the 400-kiloton warhead on the Pershing 1A. The...

...system guides the re-entry vehicle to a position within the radar range of the target area, at which time Goodyear Aerospace's terminal guidance system takes over. This system uses computer-stored digitized maps of land features of the area around the target. The radar system scans the area below, with the returns compared by the radar area correlation system with features on the prerecorded digitized maps. Steering commands are then issued to correct the re-entry path so the warhead will directly impact the target. The Pershing II uses truck-towed launcher-erector vehicles that allow the missile to be...

...Europe by 1988, while the USSR has about 243 SS-20s deployed against

European NATO targets .
19841201

4/3,KWIC/6 (Item 3 from file: 80)
DIALOG(R)File 80:TGG Aerospace/Def.Mkts(R)
(c) 2005 The Gale Group. All rts. reserv.

01037379 Supplier Number: 39381590
NAVY DEVELOPING RAPID STRIKE PLANNING
Aviation Week & Space Technology, v121, n3, p49-54
July 16, 1984
Language: English Record Type: Abstract
Document Type: Magazine/Journal; Academic

ABSTRACT:

...missions and format the data necessary for various launch platforms.
Mission planing funds are also targeted for a rapid retargeting system, a
Tomahawk employment command and control strike warfare system and...

...in the programming of a cruise missile mission include: altitude; ground
track; terrain contour matching map sets; digital scene matching area
correlation update reference sets; Mach speed; time of arrival; climb rate
and dive rate; flight mode...

...Other programmed parameters include estimation of navigation error
corridors, likelihood of a clobber, probability of map and scene update
success, likelihood of attrition and success, and fuel consumption. Terrain
countour matching compares measured altitude terrain data acquired by the
cruise missile as it flies to its target with certain flight segments and
an altitude map stored on board the missile.

The Navy has also requested \$56.9 million for de ...
19840716

4/3,KWIC/7 (Item 4 from file: 80)
DIALOG(R)File 80:TGG Aerospace/Def.Mkts(R)
(c) 2005 The Gale Group. All rts. reserv.

01033358 Supplier Number: 39349131
TOMAHAWK -- TECHNICAL
Proceedings of the US Naval Institute, v110, n975, p154-156
May, 1984

Language: English Record Type: Abstract
Document Type: Magazine/Journal; Trade

ABSTRACT:

...a vertical launcher system, or from torpedo tubes of SSN attack submarines. Terrain contour matching maps are stored in the missile and provide to-target guidance. Targeting is aided by the digital scene matching area correlator, especially in the final flight phase.

Seasonal changes can be stored in the missile's...

19840501

4/3,KWIC/8 (Item 5 from file: 80)

DIALOG(R)File 80:TGG Aerospace/Def.Mkts(R)

(c) 2005 The Gale Group. All rts. reserv.

01006304 Supplier Number: 39150377

CRUISE MISSILES: A FAMILY OF WEAPONS

Flight International, v122, n3832, p1113,111400

Oct 16, 1982

Language: English Record Type: Abstract

Document Type: Magazine/Journal; Trade

ABSTRACT:

...a cruise missile guidance system. As the weapon flies over the TERCOM field, the altimeter maps the ground and compares it with the stored matrix (based on US Defense Mapping Agency). Nuclear-armed ALCM, SLCM, and GLCM missiles use similar guidance concepts. Land attack and air launched cruise missiles use DSMAC (Digital Scene-Matching Area Correlation) that employs real-time TV imaging with a digitally stored photograph of the target or area close by. The antiship SLCM uses inertial navigation with an active radar terminal...

19821016

?

T S8/FULL/5

8/9/5 (Item 5 from file: 275)

DIALOG(R)File 275:Gale Group Computer DB(TM)

(c) 2005 The Gale Group. All rts. reserv.

01492814 SUPPLIER NUMBER: 11658116 (THIS IS THE FULL TEXT)

**Geographics. (using geographic information systems for market research)
(Computers/Communications)**

Churbuck, David

Forbes, v149, n1, p262(4)

Jan 6, 1992

CODEN: FORBA ISSN: 0015-6914 LANGUAGE: ENGLISH RECORD TY
FULLTEXT; ABSTRACT

WORD COUNT: 1074 LINE COUNT: 00084

ABSTRACT: Industry experts predict that the use of geographic information systems (GIS) will expand greatly in the 1990s because the systems are a valuable marketing tool. The systems, which are digitized maps displayed on computer screens, can be linked with databases of customer information and other other statistical data so that users can pinpoint detailed customer information for specific groups of people. Analysts believe some of the growth of GIS systems will come from the US Census Bureau's sale of economic and population data, as well as street maps. In addition, growth will come as programmers learn to combine GIS spatial databases with traditional numeric and alphabetic databases.

TEXT:

FOR A GOOD demonstration of the power of that discipline called "geographic information systems," look no further than Operation Desert Storm. In the Gulf war, everything from supply logistics to cruise-missile targets was based on computer mapping systems from Integraph Corp. and a former division of McDonnell Douglas Corp.

Now the practitioners of this art are selling their systems as a commercial weapon. Put your stores and distribution centers on a map overlaid with census tract income figures, they say, and you can see where to expand. Put a map of suburban homes into a computer and it will find the optimum layout of newspaper delivery routes. Combine fire-hydrant locations and crime statistics on a digital map and you can price homeowner insurance more competitively.

The professionals call this by its acronym, GIS. But a better name would perhaps be geographics, since the power of these digitized maps comes through only when they are displayed on a color graphics terminal. There's a huge difference between a stack of printed customer names and addresses

and a color-coded map showing where they are clustered.

"Business people are not used to looking at data on a map," says Joseph Francica, a senior manager at Integrgraph, a Huntsville, Ala. producer of hardware and software used in geographics." Business people are accustomed to looking at data in tabular formats. GIS gives them the choice between hundreds of pages of printed output or one map."

Several factors combine to give geographics potentially explosive growth over the next decade. One is that the Bureau of the Census is now selling street maps and economic and population data, detailed down to city blocks, on compact disks. A 44-disk set of street maps covers the entire U.S. for only \$11,000; 17 disks of population and economic data cost an additional \$1,700.

Another factor is that programmers are figuring out how to mesh this spatial database with traditional alphabetic and numeric databses, such as lists of customers, registered car owners and mortgagors. The marketers know who you are. Now it is only a matter of time before they know where you are.

"Most corporate data already contain some geographic elements in the form of street addresses and zip codes," says David Radoff, a spokesman for Strategic Mapping, a San Jose, Calif. maker of mapping software. "The problem is how to look at that information spatially. A report can't determine that one zip code is contiguous to another. A map can."

Geographics is an outgrowth of an older industry that provided computerized maps to civil engineers trying to keep an eye on subdivisions and manhole covers. But the next generation of customers will be very different: banks, insurance companies, department stores and junk mailers merging their customer databases into digital maps.

Arby's, the roast beef sandwich chain, uses geographics to site new stores. Chemical Bank uses it to ensure that it lends money fairly in poor neighborhoods. For insurance companies, Datamap, of Eden Prairie, Minn., provides software that can locate an address--say, 123 Main Street, Pensacola--by searching its database of digital Pensacola maps, locating the 100 block on Main Street and finding a point 23% down the street. Then it measures the distance from the home address to the nearest fire hydrant, police station and body of water. An underwriter connected to an Insurance Services Office mainframe in New York can get a near-instant risk assessment.

RR Donnelley GeoSystems, based in Lancaster, Pa., sells a locator system used by travel agents to find hotels convenient to business appointments or banks trying to determine the branch closest to a new customer. GeoSystems also sells a route-plotting program for newspaper circulation departments and a database that correlates street addresses with average income levels.

It's getting easier for firms like this to track individuals and their habits. A recent Supreme Court ruling held that phone directories are not covered by copyright. Thus, marketers can buy compact disks of phone listings--or, if necessary, scan names in from printed directories--and massage the data at low cost. James Hilliard, director of marketing and business development at GeoSystems, foresees a world in which a homeowner could quickly locate the nearest plumber or a florist could compile a mailing list of everybody within a specific radius.

Tydac Technologies Corp. in Arlington, Va. produces software (also sold by IBM with its hardware) that helps businessmen visualize competitive data. "In mapping there are three basic axes, X for latitude, Y for longitude and Z for elevation. Substitute income or some other demographic measure for elevation, and you have a commercial GIS," says Terance Moloney, a product manager at Tydac.

A typical geographics assignment is to produce a "retail gravity" model, which weights demographic data according to the needs of a particular vendor. A seller of foreign auto parts, for instance, would weight his analyses toward owners of foreign cars, as gleaned from tapes bought at the motor vehicle department.

MapInfo Corp., a Troy, N.Y. company, sells mapping software for PCs. Says Michael Marvin, chief executive, "Quaker Oats found that 80% of their ethnic customers lived in 18 of their 55 sales territories with our system. Now, by visualizing their data, they can pinpoint their in-store and direct marketing efforts."

Environmental Systems Research Institute of Redlands, Calif. sells a system used by oil, lumber and mining companies to track geophysical formations. But now the firm is aiming at a broader market. "What we hope to do with this product is [give] American business a spatial spreadsheet," says Jack Dangermond, president of the firm. Spatial information will leave the desks of trained geographers and flow through corporations the way bar charts did when spreadsheets such as Lotus 1-2-3 caught on in the 1980s, he says.

"The GIS business is at the point where the Wright Brothers were at Kitty Hawk," says Charles Foundyller, president of Daratech, a market research firm in Cambridge, Mass. "We know it works, and we know there must be a market for it somewhere, but can we imagine a 747 of an application?"

COPYRIGHT 1992 Forbes Inc.

SPECIAL FEATURES: illustration; photograph

DESCRIPTORS: Maps; Graphics Software; Marketing Research; Growth;

Marketing Strategy; New Technique; Outlook; Geographic Information System

FILE SEGMENT: MI File 47

?

8/9/4 (Item 4 from file: 275)

DIALOG(R)File 275:Gale Group Computer DB(TM)

(c) 2005 The Gale Group. All rts. reserv.

01525975 SUPPLIER NUMBER: 12344522 (THIS IS THE FULL TEXT)

A Windows assert() with symbolic stack trace. (includes related article on sanity checks for .sym files) (Tutorial)

Pietrek, Matt

Windows-DOS Developer's Journal, v3, n7, p49(10)

July, 1992

DOCUMENT TYPE: Tutorial ISSN: 1059-2407 LANGUAGE: ENGLISH

RECORD TYPE: FULLTEXT; ABSTRACT

WORD COUNT: 3847 LINE COUNT: 00299

ABSTRACT: Assertions are vital to writing robust and correct code. The assert() macro in C and C++ allows the programmer to place assertions in the code and switch assertion checking on or off before compiling. One problem with assert() is that the low-level information provided when the assertion fails is often of little direct use. Instruction is given for building a version of assert() for Microsoft Windows programs; this version supplies a stack trace with sufficient symbolic information so that the programmer can see what was transpiring when the assertion failed. The key to this version of assert() is the .sym file, which makes symbolic stack tracing possible. Many debugging tools use .sym files. A programmer performing a symbolic stack trace needs a function that, given the address of a calling function in the stack and a .sym file, returns the symbol associated with that address.

TEXT:

Assertions are an important technique for writing robust, correct code. Some languages, such as Eiffel, even have built-in support for assertions. C and C++ supply assertions in the form of the assert() macro, which allows you to place assertions in your code and switch assertion checking on or off at compile time.

One problem with assert() is that the information assert() supplies when the assertion fails is often too low-level to be of much direct use. For example, suppose your application has its own memory-management package and the function for releasing memory looks like this:

```
void myfree(void *Ptr) { assert(Ptr != NULL); /* ... */ }
```

The assertion will detect an erroneous attempt to free a NULL pointer. Unfortunately, the information assert() prints out looks something like

this:

File "myfree.c", Line# 8 Assertion failed: Ptr != NULL

Of course, what you really need to know is who passed myfree() the NULL pointer, and your application has probably called this low-level function from many different places. Thus, although assert() is good at detecting errors, you often end up having to load the application into a debugger, execute to the failed assertion, and then examine a stack traceback to locate the real culprit.

Wouldn't it be nice if assert() could also supply a stack trace with enough symbolic information for you to see what was going on when the assertion failed? This article tells you how to build a version of assert() that does just that for Windows programs. As part of the process, I describe how to read and use .sym files to provide symbolic annotation (where possible) for each frame in the stack trace. In addition, I show you how to use the new toolhelp.dll to walk the stack, saving you from having to write complex, error-prone code to walk the stack yourself. Microsoft made toolhelp.dll available with Windows 3.1, but it also works with Windows 3.0.

Design Goals

Six objectives underlie the choices I made in designing this improved assert() package.

First, the implementation should be as similar as possible to the standard C assert() macro. Code that uses assert() should be able to use the improved w[underscore]assert() with just a simple name change.

Second, there should be as little intrusion as possible into the code and makefile of the project. To use w[underscore]assert(), you only need to include one header file, and link with one library. By putting the code in a library, you prevent the linker from pulling in the w[underscore]assert() code if it is not used. If w[underscore]assert() were a .obj file, then you would have to change the link command line to prevent the w[underscore]assert() code from always being linked in.

Third, I wanted to avoid the need for an import library. w[underscore]assert() uses functions in toolhelp.dll. If w[underscore]assert() called the TOOLHELP functions directly, you would have to include the TOOLHELP import library every time you linked a program that used w[underscore]assert(). To avoid this problem, the package uses LoadLibrary() to load toolhelp.dll as needed. w[underscore]assert() calls the TOOLHELP functions through function pointers that it obtains by calling GetProcAddress().

Fourth, it should be easy to change the format of the stack trace information that w[underscore]assert() displays. To this end, the package contains an information output function that is called with a string containing information to be displayed. A second parameter tells the output

function which of three possible states it is in:

- * This is the start of information for this stack trace.
- * This is a continuation of the stack trace.
- * This is the end of the stack trace.

This lets you change the format of the stack trace without going into the internals of the stack-walking code. For instance, you could change the output routine to put the display strings in a list box and pop up the list box after receiving the last string.

Fifth, the code for accessing the .sym file should be relatively portable and not tied to the rest of the code. You should be able to take the .sym file code and use it in another project, without having to drag along any of the other `w[underscore]assert()` code. In fact, an ambitious programmer might make a C++ class or Turbo Pascal object out of it!

Sixth, and finally, the code should do appropriate error checking, but not be paranoid. This is especially true in the .sym file module. The code makes a couple of checks in the beginning to make sure it has a valid .sym file. Beyond that, however, the code does not check to make sure that every file seek or read succeeded, since that would have made it more difficult to see the underlying mechanics of reading the .sym file.

.sym Files

The key to creating this enhanced `assert()` is the .sym file, the "secret sauce" that makes symbolic stack traces possible. Although not many people are aware of them, .sym files are used by many debugging tools. Under DOS, they are used by SYMDEB. Under Windows, they are used by the Windows Debugging Kernel, HEAPWALK, WINSPCTR (a utility available in Borland C++ v3.1), DRWATSON, WDEB386, and other programs. Microsoft ships .sym files for the three Windows "core" DLLs (USER, KERNEL, and GDI) in the SDK. For your own programs, you can create .sym files by running Borland's TMAPSYM on Borland-style .map files, or by using Microsoft's MAPSYM with Microsoft-style .map files. If you do not have the source for a program or DLL, you can still create .sym files for them by using BUILDSYM, from Borland C++ v3.1.

If you are familiar with more advanced debug information formats - such as those TDW and CVW use - you will see that their relative simplicity is both the advantage and the disadvantage of .sym files. A .sym file can give you at most three things:

- * A correlation between an address and a symbol name.
- * A correlation between a code address and its source file and line number.
- * A correlation between a constant and a symbolic name for the constant.

What is missing from .sym files is symbolic type information. For instance, a .sym file cannot indicate that a symbol is an int, as opposed

to a float, or whether a symbol in a code segment is a function or a label. Because there is no way to indicate type information, user-defined data types (that is, structures) cannot be encoded. Thus, with only a .sym file to work from, you cannot ask your debugger to inspect a structure. To add to the list of negatives, .sym files contain no information about local variables, because there is no constant address for BP-based variables.

Having beaten up on .sym files, I will now praise them for their relative simplicity. Compared to the contortions required to read a TDW or CVW symbol table, reading .sym files is a dream. And, for problems such as printing stack tracebacks or symbolic disassembly, the information in a .sym file is perfectly adequate. Also, creating a .sym file is typically much faster than having your compiler produce gobs of symbol table information, and then waiting for the linker to process it all.

Until recently the .sym file format has been shrouded in mystery. With the advent of Windows 3.1, Microsoft has formally documented it in the Windows online help, under the topic "FILE FORMATS." Additionally, TDUMP (from Borland C++ or Turbo Pascal) can display both 16- and 32-bit .sym files, if you are inclined to go snooping.

I recommend you read the .sym file documentation in the Windows 3.1 SDK, since I will refer to structure names and members mentioned in that documentation. Also, for the purposes of this article, I assume the goal of reading the .sym file is to extract a symbolic name from it, given an address. That lets me ignore the information in the .sym file related to constants and line numbers and simplify the example code.

Translating Addresses to Symbols

To perform a symbolic stack trace, you need a function that, given a "logical" address (the address of a calling function in the stack) and a .sym file, returns the symbol associated with that address. As I discussed last month in "Designing a Windows Debugger," a Windows logical address is the combination of the module identifier, the order of the segments in the executable file, and the offset within the segment. Locating

Sanity Checks for .sym Files

Although the main use of the MAPDEF structure is to find the linked list of SEGDEFs, the MAPDEF structure is also important for sanity checking. Sanity checking is necessary to make sure the file really is a .sym file. Many file formats store a unique pattern in one or more beginning bytes, just to help programs identify the file type correctly. Unfortunately, the .sym file format contains no such ID bytes. Worse, Borland's precompiled headers (files with a proprietary format unrelated to Microsoft .sym files) also use the .sym extension, making it more likely that a user might accidentally pass you a bogus .sym file.

Despite these problems, you can do some rudimentary sanity checking with some of the fields in the MAPDEF structure. The first is the bFlags

field. Bit zero of this field is 0 if the file contains 16-bit symbols and 1 if the file contains 32-bit symbols; bit one of this field is 1 if the file includes an alphabetic symbol table (.sym files contain an array of pointers to symbols sorted by their associated memory address, but newer .sym files may contain an additional, alphabetically sorted, array of pointers to symbols). Since I am not expecting 32-bit symbols, I check to see that the bFlags field is equal to either 0 or 2.

Another sanity check derives from the fact that .dll or .exe files can have a maximum of 255 segments. Therefore, the MAPDEF cSegs field (the count of SEGDEF records) must be less than 256. The MAPDEF field pSegEntry contains the number of the segment containing the entry point, so it must also be less than 256 in a valid .sym file.

As a final check, I use the ppNextMap MAPDEF field to locate and read in the LAST[underscore]MAPDEF record. The first word in that structure must always be zero, so it provides a final check for a valid .sym file.

the symbol in the .sym file that corresponds to a particular logical address is fairly simple, with only a few caveats.

Figure 1 gives a rough overview of the .sym file format, showing the important parts you have to access to translate a logical address into a symbol. Where Figure 1 shows pointers, the file will contain file offsets (you must read the SDK documentation carefully, since some of the file offsets are relative to the beginning of the file and some are relative to a specific record in the file). Even more confusing, some of the offsets are "paragraph pointers," which means you must first multiply the offset by 16 to obtain the correct position in the file.

Every .sym file starts with a MAPDEF structure (see Listing 1). The MAPDEF, and the information it indexes to, corresponds roughly to one .exe or .dll. At its highest level, the .sym file is a linked list of MAPDEFs. However, I personally have never seen a .sym file containing more than one MAPDEF, not counting the last MAPDEF, which is really just an "End of the chain" marker.

The important fields in the MAPDEF structure are the SEGDEF count (cSegs) and the offset to the first SEGDEF record (ppSegDef). Symbol information in a .sym file is organized on a per-segment basis. Every segment in the .exe or .dll that appears in the .map file has a corresponding SEGDEF record in the .sym file. Given the offset of the first SEGDEF record, and the total number of SEGDEF records, you can iterate through each segment definition, looking for the desired address or symbol.

Remember that the goal of reading the .sym file was to find the symbol that corresponds to a particular logical address. The MAPDEF record is always first in the .sym file and it points to the first in a linked list of SEGDEF records. A reasonable first step, then, is to chain through the linked list of SEGDEF records is easy: just seek to and read in the

first one, then use the `ppSegDef` field (multiple it by 16 first, since the "pp" stands for "paragraph pointer") to locate the seek offset of the next `SEGDEF`.

At this point, I have to discuss a glaring omission in the Microsoft documentation - nowhere does it tell you "which" segment in the `.exe` or `.dll` the `SEGDEF` corresponds to. I am not referring to the segment name, which is given at the end of the `SEGDEF` record. What you need is the logical segment number, as given in the `.map` file. As it turns out, the segment number is stored as part of the `SEGDEF` record, but it is in a field called `wReserved1` in the Microsoft documentation. In `symfile.h` (Listing 1), I renamed the field to `segNumber` to more accurately convey its meaning.

Once you know where the logical segment number resides, you can quickly skip through the `SEGDEF` linked list, looking for a `SEGDEF` record that corresponds to the logical segment number in the logical address. If you manage to iterate through all of the `SEGDEFs`, and still have not found a `SEGDEF` that matches the desired segment, then the information just does not exist in this `.sym` file.

Once you find the correct `SEGDEF`, you can search for the symbol (`SYMDEF`) whose offset in that segment most closely corresponds to the offset of the address you are looking for. A `SYMDEF` record contains the value of a symbol (the offset portion of its logical address), a length byte, and a variable-length (up to 255) array of characters for the symbol name. A `SYMDEF` structure definition looks like this:

```
typedef struct { WORD wSymVal; BYTE cbSymName; char achSymName; /* 1s  
of array */ } SYMDEF;
```

As Figure 1 shows, the `SEGDEF` does not point directly to the `SYMDEF` structures, but to an array of pointers (offsets from the `SEGDEF` record) to `SYMDEF` structures. The reason for this extra level of indirection is to provide a convenient index for locating each variable-length symbol definition. The entries in this array are sorted by the symbols' logical addresses, so you could use a binary search to search the table more quickly for the desired address.

`findsym.c`

`findsym.c` (Listing 2) contains the code to handle the process I just described. The public interface to the code is via `FindSymbol()`:

```
void FindSymbol ( char *buffer, FILE *symfile, unsigned short seg,  
unsigned short off);
```

Given a handle to an open `.sym` file and a logical address (split into the segment and offset portions), `FindSymbol()` locates the corresponding symbol name and copies it into `buffer`. If no matching symbol is found, `FindSymbol()` sets `buffer[0]` to 0.

`FindSymbol()` uses `ReadAndVerify-SymFileHeader()` both to fetch the `MAPDEF` record and to verify that the file really is a `.sym` file (see

sidebar). After reading the MAPDEF, FindSymbol() calls PositionToCorrect-SEGDEF(), which either seeks to the SEGDEF record of the desired logical address or returns a failure code.

After positioning the file pointer to the correct SEGDEF, the last remaining job for FindSymbol() is to find the symbol nearest to the offset of the logical address by calling GetSymbolNameFromSEGDEF(), passing it the .sym file pointer, a character string buffer to fill in with the symbol name, and the offset in the segment to look for. GetSymbolNameFromSEGDEF() uses the "offset array" to perform a linear search of the SYMDEF records. This array is sorted in ascending address order, so the function simply looks for a SYMDEF with a value greater than the desired offset, and then backs up to the previous SYMDEF record.

If the first SYMDEF record has an offset greater than the desired offset, then it is a lost cause, and the function simply returns an empty string. Otherwise, GetSymbolNameFromSEGDEF() uses sprintf() to form a printable symbolic address such as:

```
[underscore]MyFunc + 002B
```

Note that the address includes the distance in bytes between the symbol address and the target offset. This is often important information, as the greater the distance between the two values, the more suspect the symbol name is. For instance, if the target offset was in a static function, then the found symbol name would be for the closest previous "public" symbol. This is because the linker will only output public symbols to the .map file.

```
W[underscore]assert.c
```

Listing 3 (w[underscore]assert.h) contains the interface definitions for the routines in w[underscore]assert.c (Listing 4). The core routine in w[underscore]assert.c is the [underscore]w[underscore]assertfail() function, which is called by the w[underscore]assert() macro. The first action undertaken is to duplicate the functionality of assert(). The standard arguments that assert() passes are sent to the generic output function [underscore]w[underscore]assertfail[underscore]output(), which I discuss later.

TOOLHELP handles most of the work of walking through the stack and extracting the return address of each function on the stack. I use StackTraceCSIPFirst() to fetch the first stack frame and StackTraceNext() to fetch subsequent stack frames (you may want to review the Windows 3.1 SDK documentation for these functions). To meet the goal of avoiding the need for import libraries, [underscore]w[underscore]assertfail() accesses these TOOLHELP functions dynamically. It does this by calling LoadLibrary(), and then calling GetProcAddress() to obtain the entry points of StackTraceCSIPFirst() and StackTraceNext().

When you call StackTraceCSIPFirst(), you have to pass it some

information about the stack frame you want to fetch. Specifically, you have to supply the SS, BP, CS, and IP registers. I use the Borland C++ pseudoregisters ([underscore]SS, [underscore]CS, and [underscore]BP) to pass these registers to StackTraceCSIPFirst(), but a bit of inline assembler could perform the same job in other compilers. It is a minor annoyance to come up with the correct value for IP in C, so I just pass 0. I can get away with that because TOOLHELP simply copies these register values to the STACKTRACEENTRY structure that it returns. Since I plan to "throw away" the first stack frame anyway, it won't matter if it contains an incorrect value for IP. It is unlikely that users will want to see the stack frame for [underscore]w[underscore]assertfail() in their stack trace.

After fetching and ignoring the first stack frame, StackTraceNext() retrieves the subsequent stack frames. The STACKTRACEENTRY that StackTraceNext() initializes contains all the information needed to call FindSymbol() and obtain a symbolic address. For each STACKTRACEENTRY, I call the local function GetSymbolicName() to handle the details of calling FindSymbol(). To assist GetSymbolicName() in performing its duties, I also send it the logical address of the program counter for the stack frame. Although TOOLHELP only really had to provide the CS:IP for the stack frame, it goes the extra mile, and provides the module handle and the logical segment corresponding to the CS value. This allows GetSymbolicName() to construct a logical address, which is necessary for looking up symbolic names. Each string that is returned from GetSymbolicName() is passed along to [underscore]w[underscore]assertfail[underscore]output().

After iterating through all the stack frames, [underscore]w[underscore]assertfail() cleans up by calling FreeLibrary() for toolhelp.dll. Almost finished, it then calls [underscore]w[underscore]assertfail[underscore]output(), telling it that there is no more output to come. Finally, [underscore]w[underscore]assertfail() calls abort() to terminate the program.

Earlier, we mentioned the GetSymbolicName() function. We'll now look at it in more detail. GetSymbolicName() takes the module handle that is passed, and uses GetModuleFileName to determine the complete path to the EXE/DLL that the module was loaded from. Because .sym files are supposed to be in the same directory as their associated EXE/DLL, GetSymbolicName() takes the complete path to the EXE/DLL, and runs it through [underscore]fnsplit() and [underscore]fnmerge(), substituting ".sym" for the file extension. If a file with the resulting name can be opened (via fopen()), then the file pointer, along with the logical segment/offset, is sent to FindSymbol(), which is in the findsym.c module. Upon returning from FindSymbol(), the .SYM file is closed, and an appropriate output string is created via sprintf().

The last piece of `w[underscore]assert.c` is the `[underscore]w[underscore]assertfail[underscore]output()` function. As implemented here, it simply calls `OutputDebugString()` to dump the string to wherever Windows is sending debug messages. Remember that, by default, `OutputDebugString()` sends its output to a debugging terminal via COM1. You can either redirect its output to a file by placing an `Output-to=filename` statement in the `[debug]` section of your `system.ini` file, or use Microsoft's DBWIN utility to view the output in a window.

Using `w[underscore]assert()`

Once you have built `w[underscore]assert.lib` (from the supplied makefile `w[underscore]assert.mak` in Listing 5), you will find it very easy to use. In any place where you would ordinarily include `assert.h` and use `assert()`, use `w[underscore]assert.h` and `w[underscore]assert()`. When you link, simply add `w[underscore]assert.lib` to the library list for the linker. You could even add `w[underscore]assert.lib` to the standard libraries supplied by your compiler vendor. You can always compile a version of your program without assertion checking just by defining the macro `NDEBUG`, either before you include `w[underscore]assert.h` or on the command line of your compiler.

Don't forget that, to really see the benefit of `w[underscore]assert()`, you have to have `.sym` files for the modules that show up in the stack trace. Install or build the `.sym` files for the Windows kernel, and use `MAPSYM` or `TMAPSYM` to build `.sym` files for your own projects.

Summary

Debugging tools and methods are often overlooked because they are viewed as being "obscure." However, as you can see from this article, the techniques for building your own tools are not that far removed from "mainstream" programming. I hope this article will inspire you to further investigate the uses of `.sym` files, and to use all of the available debugging techniques.

COPYRIGHT 1992 R&D Publications Inc.

SPECIAL FEATURES: illustration; program; chart

DESCRIPTORS: Tutorial; Programming Instruction; Type-In Programs; Program Testing Software; Stacks; Systems Programming; Trace Routines; Object-Oriented Languages; C Programming Language

TRADE NAMES: Microsoft Windows (GUI)

OPERATING PLATFORM: MS Windows

FILE SEGMENT: CD file 275

?

8/9/2 (Item 2 from file: 275)

DIALOG(R)File 275:Gale Group Computer DB(TM)

(c) 2005 The Gale Group. All rts. reserv.

02081032 SUPPLIER NUMBER: 19540809 (THIS IS THE FULL TEXT)

Functional design of the HP PA 7300LC processor. (PA-RISC 7300LC) (includ related article on timing flexibility) (Product Information)

Johnson, Leith; Undy, Stephen R.

Hewlett-Packard Journal, v48, n3, p48(13)

June, 1997

ISSN: 0018-1153 LANGUAGE: English RECORD TYPE: Fulltext; Abstract

WORD COUNT: 9956 LINE COUNT: 00764

ABSTRACT: HP designers were able to produce the low-cost PA-RISC PA-7300L microprocessor without sacrificing performance speed because of an early development focus on the memory and I/O controller (MIOC) and the CPU core. The micro-architectural advances of the PA-7300LC include an enhanced on-chip cache design, a superscalar CPU core, better configurability, frequency increases and improved linking of the main memory controllers and second-level cache. The innovative MIOC design offers improved features such as memory buffers for better utilization of graphics bandwidth, reduced copying-request latency and more efficient DMA accesses. The MIOC's design also provides improved support for dual memory widths, standard memory components, second-level cache sizes and optional error corrections scheme.

TEXT:

Microarchitecture design, with attention to optimizing specific areas of the CPU and memory and I/O subsystems, is key to meeting the cost and performance goals of a processor targeted for midrange and low-end computer systems.

The PA 7300LC microprocessor is the latest in a series of 32-bit PA-RISC processors designed by Hewlett-Packard. Like its predecessor, the PA 7100LC,(1,2) the PA 7300LC design focused on optimizing price and performance. We worked toward achieving the best performance possible within the cost structures consistent with midrange and low-end systems. This paper describes the microarchitecture of the two main components of the PA 7300LC: the CPU core and the memory and I/O controller (MIOC).

CPU Core Microarchitecture Design

Approximately one-half of the engineering effort on the PA 7300LC processor was dedicated to the design of the CPU core. The CPU core includes integer execution units, floating-point execution units, register

files, a translation lookaside buffer (TLB), and instruction and data caches. Fig. 1 shows a block diagram of the CPU core.

(Figure 1 ILLUSTRATION OMITTED)

Core Design Objectives

The design objectives for the PA 7300LC processor were to provide the best possible performance while choosing the proper set of features that would enable a system cost appropriate for entry-level and high-volume workstation products. To reach this goal, we integrated large primary caches on the processor chip and developed a tight coupling between the CPU core and the memory and I/O subsystems. The design objectives for the PA 7300LC are discussed in detail in the article on page 43.

CPU Core Differences

The PA 7300LC CPU core is derived from the PA 7100LC CPU design.(1,2) Although the PA 7300LC has many similarities with its predecessor, there are some key differences in the design that allowed us to meet our performance objectives. The first difference is that the PA 7300LC runs at 160 MHz compared to only 100 MHz for the PA 7100LC. The most obvious difference is the large primary instruction and data caches integrated directly onto the PA 7300LC chip. The PA 7100LC only has a small (1K-byte) instruction cache on the chip. Also, the organization of the caches was changed to avoid many of the stall cycles that occur on the PA 7100LC. The cache organization is discussed later in this article. The PA 7300LC has a 96-entry TLB, compared to 64 entries on the PA 7100LC. Finally, the PA 7300LC has a four-entry instruction lookaside buffer (ILAB) while the PA 7100LC's ILAB contains one entry.

Pipeline and Execution Units

Like all high-performance microprocessors, the PA 7300LC is pipelined. What is notable about the PA 7300LC pipeline is that it is relatively short at six stages, while running at 160 MHz.

Fig. 2 shows a diagram of the PA 7300LC pipeline. It does not differ greatly from the pipelines used in the PA 7200, PA 7100LC, or PA 7100 processors.(1,2,3,4) The following operations are performed in each stage of the pipeline shown in Fig. 2:

(Figure 2 ILLUSTRATION OMITTED)

1. Instruction addresses are generated in the P stage of the pipeline.
2. The instruction cache is accessed during the F stage.
3. The instructions fetched are distributed to the execution units during the first half of the I stage. During the second half of the I stage, the instructions are decoded and the appropriate general registers are read.
4. The integer units generate their results on the first half of the B stage. Memory references, such as load and store instructions, also generate their target address during the first half of the B stage.

5. Load and store data is transferred between the execution units and the data cache on the second half of the A stage.

6. The general registers are set on the second half of the R stage.

Superscalar Processor. The PA 7300LC is a superscalar processor, capable of executing two instructions per pipeline stage. This allows it, at 160 MHz, to execute at a maximum rate of 320 million instructions per second. This, however, is a peak rate that is rarely achieved on real applications. The actual average value varies with the application run. The theoretical maximum assumes the proper mix of instructions, but not every pair of instructions can be bundled together for execution in a single cycle. Fig. 3 shows which pairs of instructions can be bundled for execution in a single pipeline stage.

(Figure 3 ILLUSTRATION OMITTED)

Delayed Branching. The PA-RISC architecture includes delayed branching.(5) That is, a branch instruction will not cause the program counter to change to the branch address until after the following instruction is fetched. Because of this, branches predicted correctly with a simple branch prediction scheme execute without any pipeline stalls. The majority of the remaining branches execute with only a single stall (see Fig. 4).

(Figure 4 ILLUSTRATION OMITTED)

Two Integer Execution Units. The PA 7300LC contains two integer execution units. Each contains an ALU (arithmetic logic unit) that handles adds, subtracts, and bitwise logic operations. Only one unit, however, contains a shifter for handling the bit extract and deposit instructions defined in the PA-RISC architecture. Since only one adder is used to calculate branch targets, only one execution unit can process branch instructions. This same unit also contains the logic necessary to calculate nullification conditions.(*). By limiting execution to only one branch or nullifying instruction per pipeline stage, we avoided a great deal of functional complexity. Finally, only one unit contains the logic to generate memory addresses. Since the data cache is single-ported, there is no need to have two memory addresses generated per cycle. In special cases, however, two integer load or store instructions may be bundled together, provided they use the same double-word address. As mentioned before, these asymmetries between the integer units prevent any two arbitrary integer instructions from bundling together. However, even with this limitation, compilers are able to take advantage of the integer superscalar capabilities of the PA 7300LC.

Multimedia Instructions. The PA 7300LC integer units implement a set of instructions first introduced on the PA 7100LC that accelerate multimedia applications.(1,2,6) These instructions allow each integer unit to perform two 16-bit adds, subtracts, averages, or shift-and-adds each

cycle. Because of superscalar execution, the PA 7300LC can execute four of these operations per cycle for a peak rate of 640 million operations per second.

Floating-Point Unit. The PA 7300LC contains one floating-point unit. Contained in this unit is a floating-point adder and a floating-point multiplier. The adder takes two cycles to calculate a single- or double-precision result. It is pipelined so that it can begin a new add every cycle. The multiplier takes two cycles to produce a single-precision result and three cycles for a double-precision result. It can begin a new single-precision multiply every cycle and a new double-precision multiply every other cycle. Divides and square roots stall the CPU until a result is produced. It takes eight cycles for single-precision and 15 cycles for double-precision operations.

Instruction Cache and ILAB

Integrating a large primary instruction cache onto the processor chip broke new ground for PA-RISC microprocessors. In the past, our processor designs relied on large external primary caches. With the PA 7300LC, we felt that we could finally integrate enough cache memory on the processor chip to allow fast execution of real-world applications. Indeed, we have integrated twice as much cache on-chip as the PA 7100LC used externally in the HP 9000 Model 712/60 workstation (i.e., 128K bytes versus 64K bytes). The integrated cache not only improves performance but also reduces system cost, since an external cache is no longer mandatory.

Primary Instruction Cache. The PA 7300LC primary instruction cache holds 64K bytes of data and has a two-way set associative organization. A set associative cache configuration is difficult to achieve with an external cache, but much more practical with an integrated cache. When compared to a similarly sized directly mapped cache, it performs better because of higher use and fewer collisions. We chose a two-way associative cache over other ways to save overhead caused by the replication of comparators and to reduce the propagation delay through the way multiplexer.

The primary instruction cache is virtually indexed and physically tagged. Because the PA-RISC architecture restricts aliasing(**) to 1M-byte boundaries, we could use a portion of the virtual address (in this case, three bits) to form the index used to address the cache. To avoid using virtual address bits would have required us either to place the virtual-to-physical translation in series with cache access (increasing the cache latency) or to implement a large number of ways of associativity (in the case of a 64K-byte cache, this would have required a 16-way set associative organization).

Data Array Requirements. The instruction cache is composed of a tag array and a data array, each containing addresses and instructions. Without

using more wires or sense amplifiers than those found in a conventional cache organization, we organized the data arrays in an unusual fashion in the primary caches on the PA 7300LC to meet two requirements.

The first requirement is for the instruction cache to supply two instructions per cycle to the execution units. Because the cache is two-way set associative, each location, or set, contains instructions corresponding to two distinct physical addresses. Thus, for any given set (determined by the instruction fetch address), there are two possible choices for the instructions being read. Each of these choices is called a group (see Fig. 5a). For speed reasons, both groups are read from the instruction data arrays simultaneously. Logic that compares the physical addresses in the tag arrays (one per group) with the physical address being fetched from the data array determines which group is selected and sent to the rest of the CPU. Since this is the normal instruction fetch operation, it must be completed in a single processor cycle.

(Figure 5a ILLUSTRATION OMITTED)

The second requirement is to be able to write eight instructions simultaneously, all to the same group. Because a write occurs as part of the cache miss sequence, it is important that the write take only a single cycle to interrupt instruction fetches as little as possible.

Fig. 5a shows the conventional method of addressing data arrays. Because of electrical and layout considerations, the upper four instructions of each eight-instruction-long cache line are kept in a separate array from the lower four instructions. Both the upper and lower arrays are addressed and read concurrently. There are four arrays in total: group 0 upper, group 0 lower, group 1 upper, and group 1 lower. The instruction fetch address sent to the instruction cache, Address(0:11), contains twelve bits. One address bit, Address(10), selects between the upper and lower arrays. The rest of the address bits, Address(0:9) and Address(11), go to all four arrays and determine which set (Set(x)) is read out of the arrays. This is accomplished with the 11-to-2048 decoders. In reality, four decoders, one for each array, would be needed, but they all connect to the same address. As discussed above, there are two possible pairs of instructions to choose from with a given address. A signal from logic called hit compare selects between the two possibilities. In the example shown in Fig. 5a, instructions 0 and 1 from group 0 are selected from the instruction cache.

This conventional approach meets our first requirement. However, it does not meet our second requirement. It cannot access all eight instructions as a single group simultaneously. This is because a cache line is located in two adjacent sets and only half of the line can be read (or more important, written) at any one time. For example, if the group 0 upper array is supplying instructions 0 and 1, it obviously cannot supply 2 and

3. The only way to solve this problem with the conventional approach is to split each array into two halves. This, however, would require twice as many wires and possibly sense amplifiers producing a sizable increase in area cost. By making a slight modification to the way the data arrays are organized and addressed, we found we could avoid this pitfall and meet both of our requirements.

Our addressing approach on the PA 7300LC is called checkerboarding. Fig. 5b shows how instructions are fetched from the instruction cache on the PA 7300LC. There are, again, four arrays: left upper, left lower, right upper, and right lower. The most significant address lines, Address(0:9), go to all four arrays, while Left(11) goes only to the two left arrays and Right(11) goes only to the two right arrays. A single address bit, Address(10), selects between the upper and lower arrays, as before.

(Figure 5b ILLUSTRATION OMITTED)

When an instruction is fetched, both Left(11) and Right(11) are set to the value of Address(11). Because of this, the operation is virtually identical to the conventional approach described above, except for one key difference: a cache line for a given group is spread across all four arrays, rather than just two. This can be seen in Fig. 5b, where the instructions corresponding to group 1 have been shaded. Each array contains pieces of cache lines from both groups in a checkerboard fashion.

Fig. 5c illustrates how checkerboarding allows simultaneous access to an entire cache line. By setting Left(11) to the group desired and Right(11) to the opposite value, all eight instructions from one group can be read or written. In the example shown in the figure, an entire cache line from group 1 is read out. Left(11) is set high, while Right(11) is set low. Address(0:9) selects which pair of sets, Set(x) and Set(x+1), are accessed. Fig. 6 lists the results of addressing the arrays with the various combinations of values on Left(11) and Right(11).

(Figure 5c-6 ILLUSTRATION OMITTED)

Instruction Cache Hit Stages. The CPU core will attempt to fetch a pair of instructions from the instruction cache every cycle during which it is not stalled. For example:

- * The instruction fetch address arrives at the instruction cache at the end of the P stage of the pipeline.

- * On the first half of the F stage, the word line decoders fire one word line to each array.

- * On the second half of the F stage, the array is read, driving its value onto the bit lines to the sense amplifiers. The way multiplexer then selects the proper pair of instructions from the sense amplifier outputs.

- * On the first half of the I stage, the instructions are driven to the execution units for decoding and execution.

Instruction Cache Miss Stages. In the case of an instruction cache

miss, which is known by the end of the F stage of the pipeline, the entire pipeline will stall. A read request for an entire cache line will then be sent to the memory controller. This request is called a copyin request. A 64-bit data path between the memory controller and the instruction cache requires a minimum of four cycles to transfer the entire cache line to the instruction cache. Four cycles are required because the memory controller can only deliver 64 bits per cycle and a cache line contains 256 bits. The memory controller will return the pair of instructions originally intended to be fetched first, regardless of the pair's position within the cache line. As each pair of instructions is returned from memory, it is written into a write buffer. The instructions can be fetched directly from this buffer before they are written to the cache, with the first pair's arrival causing the pipeline to resume execution. This capability is commonly referred to as streaming. In effect, the write buffer forms a third way of associativity. After the last pair of instructions arrive from memory, the write buffer contents are written to the cache in one cycle.

Unified Translation Lookup Table. Since the instruction fetch address is a virtual address, it must be mapped into a corresponding physical address at the same time the instruction cache arrays are being accessed. Normally, a full instruction translation lookaside buffer, or ITLB, is used to perform this function. On the PA 7300LC, as on all recent PA-RISC processors, we felt that the performance improvements achieved with a separate ITLB and DTLB (for data accesses) did not warrant the increased chip area costs. Instead, we opted for a unified TLB that performs both instruction and data translations.

Instruction Lookaside Buffer (ILAB). Because both an instruction and a data translation are required on many cycles, a smaller structure called an instruction lookaside buffer, or ILAB, is used to translate instruction addresses, while the larger unified TLB is free to translate data addresses. The four-entry ILAB is a subset of the unified TLB and contains the most recently used translations. This strategy is quite effective because instruction addresses, unlike data addresses, tend to be highly correlated in space in that they generally access the same page, a previous page, or the next page.

When an instruction address does miss the ILAB, normally because of a branch, the pipeline will stall to transfer the desired translation from the unified TLB to the ILAB. We designed in two features to mitigate these ILAB stalls. On branch instructions that are not bundled with a memory access instruction (such as a load or store), the unified TLB will be accessed in parallel with the ILAB, in anticipation of an ILAB miss. If the ILAB misses, the normal ILAB stall penalty will be reduced. The second feature we added was ILAB prefetching. Every time the CPU begins executing on a new instruction page, the TLB will take the opportunity to transfer

the translation for the next page into the ILAB. This can completely avoid the ILAB misses associated with sequential code execution.

Data Cache and TLB

We designed the data cache array to be very similar to the instruction cache arrays. Like the instruction cache, the data cache is two-way set associative, virtually indexed, and physically tagged. It is composed of three arrays:

- * A data array, which has the same checkerboard organization as the instruction cache data array

- * A tag array, which is almost identical to its instruction cache counterpart

- * A dirty bit array, which has no counterpart in the instruction cache. This array keeps track of whether a data cache line has been modified by the instruction stream.

Although organized in a way similar to the instruction cache, the data cache's internal operation and effect on the CPU pipeline are quite different. The data cache and TLB operate in the A and B stages of the pipeline. A load instruction causes a data address to be generated in the first half of the B stage. The data cache word line decoders operate on the second half of the B stage. On the first half of the A stage, the arrays drive their values out. Based on the comparison between the physical address and the output of the tag arrays, the way multiplexer then selects the proper data value. This word or double-word value is then driven to the integer and floating-point units during the second half of the A stage.

A store instruction generates a data address in the same manner as a load instruction. That address is used to read from the tag array as described above. Instead of using the store address to read from the data array, however, the address from the head of a two-entry store queue (Fig. 7) is used to index into the data array on the second half of the B stage. The data from the head of the store queue is written into the data array on the first half of the A stage. The data from the store instruction is driven from the integer or floating-point units to the data cache on the second half of the A stage where it is written into the tail of the store queue.

(Figure 7 ILLUSTRATION OMITTED)

Store Queue. A load can retrieve data directly out of the store queue if it is to the same address as the store. The necessity of the store queue is twofold:

- * The floating-point unit cannot drive store data in time to write the data array during the proper pipeline stage. The store queue, therefore, provides the time to transfer the data from the execution units to the data cache.

- * Memory cannot be modified until it is known that the store

instruction will properly finish execution. If the store instruction is going to trap, say, because of a TLB fault, any architected state, such as memory, must not be changed.

The disposition of the store instruction is not known until the R stage of the pipeline, well after the data array is to be written. The store queue serves as a temporary buffer to hold pending store data. If a store that writes into the store queue subsequently traps, that store queue entry is merely invalidated. Also, by using a store queue, we are able to use a single bidirectional bus to transfer data between the execution units and the data cache. The store queue allows data to be transferred on the second half of the A pipeline stage for both load and store instructions, preventing conflicts between adjacent loads and stores in the instruction stream.

Semaphore Instructions. The data cache performs other memory operations besides load and store instructions. It handles semaphore instructions, which in the PA-RISC architecture require a memory location to be read while that location is simultaneously zeroed. In operation, a semaphore is quite similar to a store instruction with zeroed data, except that the semaphore read data is transferred on the second half of the A stage. In cases in which the semaphore is not present or modified in the data cache, the load and clear operation must be performed by the memory controller.

Flush and Purge Instructions. We must also execute flush instructions, which cause a given memory location to be cast out of the data cache. Related is the purge instruction, which at the most privileged level causes a memory location to be invalidated in the data cache with no cast out, even if the line is modified.

Reducing Miss Latency. Data cache misses are detected on the first half of the A stage of the pipeline. To reduce miss latency, the physical address being read from the data cache is forwarded to the memory controller before the data cache hit-or-miss disposition is known. This address is driven to the memory controller on the first half of the A stage. A "use address" signal is driven to the memory controller on the first half of the R stage if a cache miss occurs.

Copyin Transaction. A number of transaction types are supported between the CPU core and the memory controller. The most common type is a copyin transaction.

After receiving a copyin request, the memory controller returns the requested cache line, one double word at a time. As with instruction misses, the memory controller returns the data double word that was originally intended to be fetched first.

On load misses, when the critical double word arrives, it is sent directly to the execution units for posting into the register files. On

integer load misses, the critical data is bypassed before error correction to reduce latency even further.

In the extremely rare event that the data contains an error, the CPU is prevented from using the bad data and forced to wait for corrected data. As each double word arrives from the memory controller, it is placed into a copyin buffer.

When all the data has arrived, the contents of the copyin buffer are written to the data cache data array in a single cycle. There are actually two copyin buffers to ensure that two data cache misses can be handled simultaneously.

Fig. 8 shows a block diagram of the copyin and copyout buffers.

(Figure 8 ILLUSTRATION OMITTED)

Copyout Transaction. A data cache line can contain modified data requiring posting or writing back to memory when cast out. To this end, another transaction type is implemented--a copyout transaction. A copyout is necessary under two circumstances. The first case is when a data cache miss is detected and the existing cache line selected for replacement has been modified. This is the most common case.

The second case is when a flush instruction is executed and hits a modified line in the data cache. The data cache supplies both a physical address and 32 bytes of data on a copyout. The data cache uses the checkerboard organization, so the full cache line read for the copyout takes only one cycle.

Reducing Cache Miss Penalties. In the PA 7300LC, we have taken a number of steps to reduce the penalty caused by cache misses. As mentioned above, we have reduced cache miss latencies. We have also continued to adopt a "stall-on-use" load miss policy pioneered on earlier PA-RISC designs.(4) In this policy a load miss stalls the CPU pipeline only long enough to issue the copyin transaction and possibly a copyout transaction. In many cases, the delay lasts for only one cycle. The CPU will then only stall when the target register of the load instruction is subsequently referenced. If the critical data returns from memory fast enough, the pipeline will not stall at all.

Because memory data is not needed by the CPU on a store miss, the CPU only stalls once, again for only one cycle in many cases, to issue the copyin and copyout transactions.

A scoreboard keeps track of which words have been stored so that the copyin write will not overwrite more recent data. Since high-bandwidth writes to I/O space can be critical to graphics performance, under most circumstances the PA 7300LC will not stall on a store to I/O space. This optimization is possible because an I/O space access is guaranteed to miss the data cache, so there is no need to stall the CPU to perform a copyout read.

Cache Hints. The PA-RISC architecture defines cache hints to allow the programmer or compiler to communicate information that can be used by the hardware to improve performance.(5) We have implemented two of these hints on the PA 7300LC:

- * Block copy store. Hints are used to indicate that software intends to write an entire cache line. In this case, there is no need to perform a main memory read on a cache miss. With this hint specified, upon detecting a store miss, the PA 7300LC simply zeros out a copyin buffer and continues without issuing a copyin transaction.

- * Coherent operation semaphore hint. This optimization improves semaphore performance by not forcing the load and clear operation to the memory unit if the data is present in the cache.

TLB Access. All memory reference instructions are guaranteed access to the unified TLB containing both instruction and data translations, during the B and A stages of the pipeline. The TLB is fully associative and contains 96 page translations. The TLB receives a virtual data address on the first half of the B stage and drives a translated physical address on the first half of the A stage. This physical address goes to the data cache to perform hit comparison and to the memory controller in anticipation of a data cache miss.

In addition to containing 96 page entries, each of which maps to a 4K-byte page, the TLB also contains eight block entries used to map larger memory ranges. These block entries are managed by the operating system.

CPU Summary

Although the CPU core of the PA 7300LC is not dramatically different from its predecessors, several noteworthy features that improve performance and allow more cost-effective system designs include:

- * A simple pipeline and a capable superscalar core that increased our operating frequency to 160 MHz.

- * Substantial primary caches integrated directly onto the processor chip

- * Most important, cache controllers that take advantage of integrated caches, resulting in features designed into the CPU core to increase the competitiveness of PA 7300LC-based systems.

Memory and I/O Controller Design

The memory and I/O controller (MIOC) is responsible for interfacing the CPU core to the memory and I/O subsystems. Integrating the MIOC on the same chip as the CPU core provides a tight coupling that results in outstanding memory and I/O performance. The memory controller includes a main memory controller and a controller for an optional second-level cache. The I/O controller interfaces the CPU core to HP's general system connect (GSC) I/O bus and handles direct memory access (DMA) requests from I/O devices.

CPU to MIOC Interface

The CPU core transmits four basic types of request to the MIOC:

- * Copyins. These requests occur during first-level cache misses and are used by the CPU core to read a cache line from the memory subsystem.
- * Copyouts. A copyout is a cache line from the CPU core that must be written to the memory subsystem because it was modified in the first-level cache by a store instruction. Copyouts are only issued when a modified cache line is replaced or flushed from the first-level cache.
- * Uncached loads and stores. An uncached load or store request is a read or write to either memory or I/O for an amount of data that is less than a cache line.
- * Load-and-clears. This request is an indivisible request to read a location and then clear it. This operation is needed to implement PA-RISC's semaphore mechanism. Requests that have addresses located in memory address space are processed by the memory controller, and all others are sent to the I/O controller.

The PA 7300LC has a four-entry copyout buffer. Copyouts are posted to memory as a background operation, allowing copyins to be processed before copyouts. New copyin requests are checked for conflict within the copyout buffer. If there is no conflict, the copyin is processed before all copyouts to help minimize load use stalls.

Second-Level Cache Control

Even though first-level caches on the PA 7300LC are relatively large for integrated caches, many applications have data sets that are too big to fit into them. The second-level cache (SLC) implemented for the PA 7300LC helps solve this problem. Logically, the SLC appears as a high-speed memory buffer; other than its performance improvement, it is transparent to software. The SLC is physically indexed, is write-through, has unified instructions and data, and is direct mapped.

The SLC becomes active after an access misses the first-level cache. The first-level cache miss indication becomes available after the TLB delivers the real address. As a result, there is little advantage to virtually indexing the SLC and real indexing avoids the aliasing problems associated with virtual caches.

Multiway Associative Cache Comparison. Multiway associative caches enjoy better hit rates because of fewer collisions. However, multiway caches are slower because of way selection, and for a given cache size, are much more expensive to implement with industry-standard components. For most applications, it is more advantageous to trade off size for ways of associativity.

Write-Back Cache Comparison. Write-back caches(*) generally have better performance than write-through caches. However, sharing the data bus with main memory alters this situation. If the SLC were write-back, lines

copied out of the SLC would have to be read into the PA 7300LC, the error-correcting code (ECC) would have to be computed, and the line would have to be written back to main memory. This operation would be quite expensive in terms of bus bandwidth. Instead, dirty lines cast out by the first-level cache are written to the SLC and to main memory simultaneously.

Any valid line in the SLC always has the same data as its corresponding location in main memory. Writing simultaneously to main memory and to the SLC is slightly slower than simply writing to the SLC SRAM, but produces a good performance and complexity trade-off when compared to a write-back design.

DMA Interface. DMA reads and writes from I/O devices are typically sequential and do not exhibit the access locality patterns typical of CPU traffic. Entering DMA traffic into the SLC tends to pollute the SLC with ineffective entries. Instead, buffering and prefetching inside the DMA interface are better ways of improving DMA performance. To maintain consistency, an SLC check cycle is run for DMA writes, and if it hits, the line is marked invalid. DMA write data is always written to main memory and DMA reads are always satisfied from main memory. Because of the write-through design of the SLC described above, data in the SLC never becomes stale.

SRAM Components. The PA 7300LC is optimized for both price and performance. Relatively early in the design process, it became necessary to select the static random access memory (SRAM) components used to build the SLC. SRAM components are frequently used in cache construction because they offer high speed with moderate cost and capacities. Given the relatively long design cycles necessary to produce a complex microprocessor and the uncertainties of the semiconductor marketplace, it was impossible to predict which components would be most attractive from a price and performance perspective when the PA 7300LC entered full production. Instead of selecting a single component, the decision was made to support a broad range of SRAM types. This allowed component selection to be made late in the development cycle and even upgraded at some point during the production phase.

Second-Level Cache Size. Most popular computer benchmark programs have relatively small working sets and are not particularly sensitive to the performance of the memory system beyond the first-level cache. On the other hand, application programs have widely variable working set sizes. Some are small and fit well in the first-level cache and some exhibit little reference locality and don't fit in any reasonably sized cache. Hence, no single SLC size is appropriate. The PA 7300LC SLC controller supports cache sizes ranging from 256K bytes up to 64M bytes. Although a 64M-byte SLC is expensive, it might be cost-effective for some applications.

The SLC data array width can be programmed to either 64 or 128 bits

plus optional ECC. However, the width must match the width of main memory.

Memory Arrays. The SLC consists of two memory arrays: the data array and the tag array. The data array shares the data bus with main memory. As an option, ECC bits can be added to the data array, and the full single-bit correct and double-bit detect error control invoked for SLC reads. The tag array includes a single optional parity bit. If parity is enabled and bad parity is detected on a tag access, an SLC miss is signaled, the failing tag and address are logged, and a machine check is signaled.

Main Memory Control

DRAMs. Dynamic random access memory (DRAM) technology is used to construct main memory because of its high density, low cost, and reasonable performance levels. The main memory controller supports industry-standard DRAMs from 4M-bit to 256M-bit capacities. Systems can have up to 16 slots and total memory can be up to 3.75G bytes, the maximum possible with the PA-RISC 1.1 architecture.

Data Bus Width. Data bus width can be either 64 or 128 bits plus optional ECC. The 128-bit data bus width significantly improves memory performance. The 64-bit option supports lower-cost systems.

Main Memory Controller. The PA 7300LC main memory controller is very flexible and is able to support most types of asynchronous DRAMs. The controller is intentionally not SIMM/DIMM (single or double inline memory module) specific. This allows use of the PA 7300LC in a wide variety of system configurations. The main memory can support extended data out (EDO) DRAMs, which are similar to other DRAMs but use a slightly modified protocol that pipelines the column access.

Fig. 9 shows the timing diagrams of read accesses, emphasizing the improved data bandwidth of EDO DRAMs compared to standard page-mode DRAM (Figure 9 ILLUSTRATION OMITTED)

Error-Correcting Code. The state of DRAM memory cells is susceptible to corruption from incident energetic atomic particles. Because of this, the PA 7300LC main memory controller optionally generates and checks an error-correcting code. The code is generated over a 64-bit data word. Any single-bit error within the 64-bit data word can be corrected. All double-bit errors and all three- or four-bit errors within an aligned nibble can be detected. The aligned nibble capability is useful since memory systems are typically built with four-bit-wide DRAMs. The nibble mode capability allows detection of the catastrophic failure of a single four-bit-wide DRAM. Whenever an error is detected, data and address logging registers are activated to support efficient fault isolation and rapid field repair.

Shared SLC and Main Memory Data Bus

From a cost perspective, it was desirable to share the large data buses needed for the SLC and main memory, thereby lowering the pin count of

the PA 7300LC. However, sharing the large load from main memory DRAM cards would have significantly impacted the speed of SLC operations. The solution to this problem resulted in using an FET switch to isolate the main memory load from the SLC bus when the SLC is driving the bus, but to allow the bus to be shared when main memory is being accessed (see Fig. 10). The FET switch is a relatively inexpensive industry-standard part, which has a propagation delay of less than 1 ns when in the on state.

(Figure 10 ILLUSTRATION OMITTED)

FET Switch. The FET switch also enabled us to connect the PA 7300LC to legacy 5-volt DRAM cards. The PA 7300LC operates at 3.3 volts and is not tolerant of the 5-volt logic swing of many existing DRAM cards. Biasing the gate of the FET switch to a voltage lower than 5 volts effectively limits the voltage swing from DRAM cards to 3.3 volts when seen by the PA 7300LC.

Chip Layout Challenges

Although the MIOC is a small part of the PA 7300LC, it controls nearly all of the I/O pins. Because the pins are located at the chip perimeter, long signal routes from the MIOC to some pins are unavoidable. Separating the MIOC into several blocks that could be placed near the chip perimeter and controlled remotely helped manage this problem. In particular, the data flow across the shared SLC and main memory data bus is completely predictable (because there are no slave handshakes from the memories), making the memory data interface the ideal block to be controlled from the other side of the chip.

Cache Miss Data Flow

The MIOC is highly optimized for satisfying CPU cache misses. Although DMA transaction processing is handled efficiently, system performance is more sensitive to CPU cache miss performance than DMA performance.

When idle, the SLC and main memory controllers pass through physical addresses that are coming directly from the TLB and going to the SLC and main memory address pads. On the cycle following each address, the CPU core indicates whether that address resulted in a miss in the first-level cache. If a miss occurred, then an access is initiated and a cycle is saved by having passed along the physical addresses to the SLC and main memory.

For copying, the SLC begins an access. The tag and data array are accessed in parallel. If there is an SLC hit, then data is returned to the processor.

On an SLC miss, the SLC data array data drivers are disabled, the FET switch is closed, and control is transferred to the main memory controller.

When a transaction is received by the main memory controller, it endeavors to activate the correct DRAM page. This may be as simple as issuing a row address strobe (RAS) with the proper row address, or may require deasserting RAS, precharge, and a new RAS. The memory controller sequences up to the point at which it is ready to issue a column address

strobe (CAS) command, waits there until the SLC misses, and switches control over to complete the CAS command. However, if the SLC hits, it will wait for the next transaction and start the cycle again. Performance is improved by starting the DRAM access in parallel with the SLC access.

In the case of an SLC miss, once the main memory controller has control, it issues the proper number of CAS cycles to read the data. As the data passes the SLC, it is latched into the SLC data array. At the end of the cycle, the FET switch is opened, the SLC drivers are enabled, and the next transaction is processed.

Reducing Low-Miss Latencies. Much of the work described above concerns reducing miss latencies. This is important because even though the PA 7300LC CPU core has a nonblocking cache, load use stalls still develop quickly for many instruction sequences. Low-miss latencies minimize the impact of these stalls, which results in better overall performance. At CPU clock rates of 160 MHz, the PA 7300LC, as seen by the CPU pipeline, is capable of SLC hit latencies of three cycles with industry-standard 6-ns asynchronous SRAM. Main memory latencies can be as low as 13 cycles with 50-ns DRAM. Many single-cycle latency reductions have been implemented in the PA 7300LC; each by itself would not have much impact on overall memory access latency, but taken together, they make a significant difference.

I/O Interface

The PA 7300LC contains interface logic that allows direct connection to HP's high-speed GSC I/O bus. This interface processes I/O requests from the CPU core and DMA requests from GSC I/O bus devices.

Programmed I/O. Programmed I/O allows load and store instructions from the CPU core to communicate with the I/O subsystem. From a performance perspective, programmed I/O writes to graphics devices are important for many workstation applications. The improvements made for graphics performance in the PA 7300LC are described later in this article.

DMA Interface Controller. The DMA interface controller is designed to minimize main memory controller traffic and to reduce DMA read latency. The DMA interface controller employs three 32-byte line buffers. When servicing any DMA read, the controller requests 32 bytes from main memory and puts the data into one of the buffers. DMA requests on the GSC bus may be 4, 8, 16, or 32 bytes long. Since most DMA requests are to sequential addresses, requests less than 32 bytes can probably be satisfied from data contained in the buffer without issuing another request to the main memory controller. The DMA controller is also able to prefetch the next sequential line of information to increase the chances that DMA read requests are serviced from the DMA buffers.

GSC Write Requests. Writes are collected by the DMA hardware and passed on to the main memory controller. GSC write requests of 32 bytes are sent directly to the controller, but when possible, smaller-sized writes

are collected into 32-byte chunks by the DMA controller to allow the main memory controller to access memory more efficiently.

Improvements for Graphics Applications

Graphics performance depends on many aspects of the system design. In addition, graphics workloads are sensitive to the system architecture. For the PA 7300LC, we chose to optimize the design for engineering graphics, where the typical workload involves rendering an object to the display device.

From a high-level point of view, the process of rendering an object can be divided into three steps:

1. Traversing the display list that describes the object
2. Clipping, scaling, and rotating the object with the current viewpoint
3. Transforming the object from primitive elements, such as polygons, into pixels on the screen.

This process can be partitioned in different ways. With today's powerful CPUs, the most cost-effective method is to store the display list in the computer system's main memory. The host CPU performs the display list traversal and the clipping, scaling, and rotation steps, and then passes primitives to dedicated graphics hardware for conversion into onscreen pixels.

Graphics Requirements. Several different models, including specialized CPU instructions and DMA engines, have been used to extract data to be rendered from main memory. While these approaches work, they incur the undesirable cost of specialized driver software that doesn't port well between processor generations. Starting with the PA 7100LC, the philosophy has been to support the graphics requirements within the existing architecture as much as possible. For example, the PA-RISC architecture defines a set of 32-bit integer unit general registers and another set of 64-bit floating-point unit general registers. Loads and stores from either set can be made to memory space, but only integer register loads and stores were architecturally defined to I/O space.

Starting with the PA 7100LC, floating-point register loads and stores to I/O space have been implemented. This has yielded improved performance because a single load or store can now move 64 bits and because more registers are available for operations that communicate with I/O space.

In contrast with specialized operations, extensions within the architecture are generally applicable and carry forward into future generations. These optimizations can also be used to benefit workloads other than graphics.

Graphics Optimizations. Several of the optimizations made in the PA 7300LC to further improve graphics performance include:

- * A large I/O store buffer

- * A relaxation of the load and store ordering rules
- * The elimination of a CPU hang cycle previously needed for I/O

stores

- * Improvements to the GSC I/O bus protocol.

The structure of industry-standard graphics libraries leads to bursty graphics I/O traffic. The bursts are of many different sizes, but the most common burst is a write of 26 words. The PA 7300LC CPU core-to-I/O interface implements a large write buffer and can accept up to 19 double-word writes without stalling the CPU pipeline. This allows the CPU core to burst up to 19 double-word writes to the I/O subsystem, and then continue with its next task while the I/O interface is sending this data out to the graphics hardware.

Graphics Ordering. PA-RISC is a strongly ordered architecture. Strongly ordered means that all elements of the system must have a consistent view of system operations. In the case of graphics performance, this means that all buffered I/O stores must be observed by the graphics device before the CPU can access a subsequent piece of data in main memory. Hence, an I/O store and a following memory read are serialized. A loophole to the ordering requirement was created for graphics. I/O stores within a programmable address range are allowed to be out-of-order with respect to the memory accesses. The graphics software takes responsibility for ordering when necessary.

Hang Cycle. Previous PA-RISC processors always incurred a minimum of one hang cycle for an I/O store. Extra logic was added to the data cache controller on the CPU core to eliminate this hang cycle.

Graphics Transfer Size. HP's high-speed GSC bus is used to connect graphics adapters to the PA 7300LC. The CPU sends data to the graphics device with I/O stores. In the PA-RISC architecture, I/O stores are 64 bits or less. The GSC is a 32-bit multiplexed address and data bus. Stores of 64 bits turn into an address cycle followed by two data cycles. At best the payload can be only two thirds of the maximum bus bandwidth. As mentioned above, the average transfer size to graphics is 26 words. Since these transfers are sequential, sending an address with every two words is unnecessary. Some form of address suppression or clustering of sequential writes was desired. Thus, the write-variable transaction was created.

Write-Variable Transactions. A new write-variable transaction type was created for the GSC bus. Write-variable transactions consist of an address and from one to eight data cycles. Since the PA 7300LC must be compatible with existing cards that do not implement the write-variable cycle type, the PA 7300LC only generates them in configurable address spaces.

With this protocol, the I/O controller blindly issues write-variable transactions for enabled I/O address regions. Starting with the initial write, as each write is retired from the I/O write queue, the I/O

controller performs a sequentiality check on the next transaction in the queue. The process repeats for up to eight GSC data cycles. Maximum performance is achieved by allowing the I/O controller to begin issuing the write when the first piece of data becomes available.

The length of the transaction is limited to eight data cycles.

Choosing eight data cycles is a good compromise between flow control issues and amortizing address cycle overhead with payload. The write-variable enhancement increased maximum CPU-to-graphics bandwidth from two thirds of the GSC raw bandwidth to 8/9 of the raw bandwidth. The PA 7300LC can easily saturate the GSC bus at 142 Mbytes per second compared with the 50 Mbytes per second achieved by the PA 7100LC with careful coding.

MIOC Summary. The MIOC implemented a number of features that improve system performance while keeping costs low, including:

- * The second-level cache and main memory controllers are optimized to reduce the latency of copyin requests from the CPU core.

- * The I/O controller improves graphics bandwidth and supports efficient DMA accesses through the use of buffers and prefetching.

- * The MIOC is designed to be flexible, supporting a range of second-level cache sizes, a variety of industry-standard memory components, two different memory widths, and an optional error correction scheme.

Conclusion

The PA 7300LC design builds on the success of past processor designs and offers significant improvements in key areas. It features a superscalar CPU core, a large, efficient on-chip cache organization, tightly coupled second-level cache and main memory controllers, and bandwidth improvements for graphics. These features combined with frequency increases, extensive configurability, and high chip quality make the PA 7300LC attractive for a wide range of computer systems.

Acknowledgments

A large number of people were responsible for the successful design of the PA 7300LC, especially the design teams from the Engineering Systems Lab in Fort Collins and from the Integrated Circuits Business Division's Fort Collins Design Center. Many important contributions were also made by individuals from the Fort Collins Systems Lab, the Systems Performance Lab in Cupertino, the Computer Technology Lab in Cupertino, and other organizations within HP.

(*) The PA-RISC architecture enables certain instructions to conditionally nullify or cancel the operation of the following instruction based on the results of the current calculation or comparison.

(**) Aliasing refers to intentionally allowing two different virtual addresses to map to the same physical address. The PA-RISC architecture restricts the number and location of bits that may differ between two virtual addresses.

(*) In a write-back cache design (also called copy-back), data is written only to the cache on a write, and is not written to main memory until the cache line is invalidated. In a write-through cache design, data is written to

References

- (1.) P. Knebel, et al, "HP's PA 7100LC: A Low-Cost Superscalar PA-RISC Processor," Proceedings of IEEE Compcon, February 1993, pp. 441-447.
- (2.) S. Undy, et al, "A VLSI Chipset for Graphics and Multimedia Workstations," IEEE Micro, Vol. 14, no. 2, April 1994, pp. 10-22.
- (3.) G. Kurpanek, et al, "PA 7200: A PA-RISC Processor with Integrated High-Performance MP Bus Interface," Proceedings of IEEE Compcon, February 1994, pp. 375-382.
- (4.) E. DeLano, et al, "A High-Speed Superscalar PA-RISC Processor," Proceedings of IEEE Compcon, February 1992, pp. 116-121.
- (5.) R. Lee, "Precision Architecture," IEEE Computer, Vol. 22, no. 1, January 1989, pp. 78-91.
- (6.) R. Lee, J. Beck, L. Lamb, and K. Severson, "Real-Time Software MPEG Video Decoder on Multimedia-Enhanced PA 7100LC Processors," Hewlett-Packard Journal, Vol. 46, no. 2, April 1995, pp. 60-68.
- (7.) M. Bass, T. Blanchard, D. Josephson, D. Weir, and D. Halperin, "Design Methodologies for the PA 7100LC Microprocessor," Hewlett-Packard Journal, Vol. 46, no. 2, April 1995, pp. 23-35.

RELATED ARTICLE: Timing Flexibility

Microprocessor design is a time-consuming and expensive process. Ideally, a design should scale through several fabrication process generations with low-investment algorithmic artwork shrunk to help amortize the cost of the original design.

Although it is relatively straightforward to increase the processor frequency, the frequency of interconnect to the rest of the system is more or less fixed. Typically the base processor design has the capability for a range of core-processor-frequency-to-interconnect-frequency ratios.

The PA 7300LC has three interfaces that are tolerant of increases in the processor frequency: the I/O bus interface, the main memory interface, and the second-level cache interface.

The cycle time of the general system connect (GSC) I/O bus can be configured to some multiple of the processor's cycle time. The I/O controller supports ratios from three to nine. The second-level cache controller can be configured to support a variable number of CPU cycles per second-level cache cycle. The controller supports two, three, or four CPU cycles per cache cycle. Similarly, the main memory controller can configure the setup and hold times of the DRAMs to be two, three, or four CPU cycles. Additionally, seven key DRAM timing parameters can be individually programmed.

As the processor gets faster, performance may improve but only as a sublinear function of processor frequency since memory and I/O performance remain constant. The large first-level caches on the PA 7300LC help insulate the processor from the effects of the relatively slow memory accesses, allowing the performance to scale well with increasing core processor frequency. The initial frequency target for the PA 7300LC was 132 MHz, but design ratios support core processor frequencies up to 360 MHz.

Two additional benefits are derived from the timing flexibility of the PA 7300LC. The increasing availability of higher-speed DRAMs and SRAMs makes it a simple matter to configure the timing generators to take advantage of these new components. Also, timing flexibility decouples the design effort from uncertainties that develop as RAM component vendors traverse their own development cycles.

Leith Johnson Leith Johnson is a member of the technical staff at HP's Systems Technology Division and is currently working on the next-generation mid- and high-end core electronics complexes. He recently worked on the memory and second-level cache controller design for the PA 7300LC CPU. Since joining HP in 1978, his memorable contributions include working on the bit-slice processor for the HP 9845 desktop computer, the processor board design for the HP 3000 Model 825 and HP 9000 Model 925 computer systems, the memory controller design for the original HP 9000 Series 700, and the memory and I/O controller design for the PA 7100LC. He is professionally interested in computer system design with an emphasis on memory controllers and is named as an inventor in five patents related to high-speed computer bus design and in two patents pending for his work on the PA 7300LC. Born in Seattle, Washington, Leith was awarded a BSEE degree in 1978 from the University of Nevada at Reno and an MSCS in 1988 from Colorado State University. In his free time, he enjoys outdoor sports such as back country skiing and bicycling and also likes to play pinball.

Stephen R. Undy An engineer/scientist in the engineering systems laboratory at HP's Systems Technology Division, Steve Undy recently worked as the microarchitect on the data cache for the HP PA 7300LC processor and is currently responsible for the instruction cache on a future microprocessor. He is professionally interested in computer architecture, caches, and microprocessor verification and coauthored several articles on these subjects for the HP Journal, CompCon conferences, and the IEEE, of which he is a member. Steve received a BSE degree in electrical engineering and a BSE degree in computer engineering, both from the University of Michigan in 1983. He then earned an MSEE degree in 1985 from Purdue University. He joined HP's Information Hardware Operation that same year and his two favorite projects since then include designing the instruction and data cache for the PA 7100LC and working on the HP 9000 Series 700 computer, designing the CPU cache control and verifying the custom memory

and I/O controller chip. Steve was born in Detroit, Michigan and is married. He enjoys outdoor activities such as sailing, scuba diving, cycling, astronomy, and woodworking. He also enjoys theater and reading.

COPYRIGHT 1997 Hewlett Packard Company

SPECIAL FEATURES: table; chart; illustration

COMPANY NAMES: Hewlett-Packard Co.--Product development

DESCRIPTORS: Microprocessor; Product Description/Specification

PRODUCT/INDUSTRY NAMES: 3674124 (Microprocessor Chips)

SIC CODES: 3674 Semiconductors and related devices

TICKER SYMBOLS: HWP

TRADE NAMES: HP PA-RISC PA-7300LC (Microprocessor)--Design and construction

FILE SEGMENT: CD File 275

?

Hit List

Clear

Generate Collection

Print

Fwd Refs

Bkwd Refs

Generate OACS

Search Results - Record(s) 1 through 10 of 11 returned.☐ 1. Document ID: US 20020090953 A1

L3: Entry 1 of 11

File: PGPB

Jul 11, 2002

PGPUB-DOCUMENT-NUMBER: 20020090953

PGPUB-FILING-TYPE: new

DOCUMENT-IDENTIFIER: US 20020090953 A1

TITLE: Communication method and communication system for controlling with limited area information

PUBLICATION-DATE: July 11, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Aburai, Maki	Kawasaki		JP	
Hirasawa, Mitsuru	Yokohama		JP	
Hamaguchi, Kazuko	Yokohama		JP	
Sasaki, Toshiichirou	Iwaki		JP	

US-CL-CURRENT: 455/456.1; 455/457

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments	Claims	KWIC	Draw D
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------	--------	------	--------

☐ 2. Document ID: US 20020026281 A1

L3: Entry 2 of 11

File: PGPB

Feb 28, 2002

PGPUB-DOCUMENT-NUMBER: 20020026281

PGPUB-FILING-TYPE: new

DOCUMENT-IDENTIFIER: US 20020026281 A1

TITLE: On-vehicle vehicle guide apparatus, communication server system, and substitute vehicle guide system

PUBLICATION-DATE: February 28, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Shibata, Makoto	Saga		JP	
Moribe, Nobuyuki	Fukuoka		JP	
Yamaguchi, Tomonari	Fukuoka		JP	

US-CL-CURRENT: 701/208; 340/995.1

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments	Claims	KWIC	Drawn De
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------	--------	------	----------

☐ 3. Document ID: US 20020004701 A1

L3: Entry 3 of 11

File: PGPB

Jan 10, 2002

PGPUB-DOCUMENT-NUMBER: 20020004701

PGPUB-FILING-TYPE: new

DOCUMENT-IDENTIFIER: US 20020004701 A1

TITLE: Server, method and program for updating road information in map information providing system, and recording medium with program recording

PUBLICATION-DATE: January 10, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Nakano, Toshiaki	Tokyo		JP	

US-CL-CURRENT: 701/200; 701/208, 709/217

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments	Claims	KWIC	Drawn De
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------	--------	------	----------

☐ 4. Document ID: US 6484091 B2

L3: Entry 4 of 11

File: USPT

Nov 19, 2002

US-PAT-NO: 6484091

DOCUMENT-IDENTIFIER: US 6484091 B2

TITLE: On-vehicle vehicle guide apparatus, communication server system, and substitute vehicle guide system

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments	Claims	KWIC	Drawn De
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------	--------	------	----------

☐ 5. Document ID: US 5948040 A

L3: Entry 5 of 11

File: USPT

Sep 7, 1999

US-PAT-NO: 5948040

DOCUMENT-IDENTIFIER: US 5948040 A

TITLE: Travel reservation information and planning system

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments	Claims	KWIC	Drawn De
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------	--------	------	----------

☐ 6. Document ID: US 5933776 A

L3: Entry 6 of 11

File: USPT

Aug 3, 1999

US-PAT-NO: 5933776

DOCUMENT-IDENTIFIER: US 5933776 A

TITLE: Method and apparatus for field testing cellular telephones

Full	Title	Citation	Front	Review	Classification	Date	Reference	Claims	KWIC	Draw. De
------	-------	----------	-------	--------	----------------	------	-----------	--------	------	----------

☐ 7. Document ID: US 5719387 A

L3: Entry 7 of 11

File: USPT

Feb 17, 1998

US-PAT-NO: 5719387

DOCUMENT-IDENTIFIER: US 5719387 A

TITLE: IC card including a memory, a password collating means and an access permitting means for permitting access to the memory

Full	Title	Citation	Front	Review	Classification	Date	Reference	Claims	KWIC	Draw. De
------	-------	----------	-------	--------	----------------	------	-----------	--------	------	----------

☐ 8. Document ID: US 5698836 A

L3: Entry 8 of 11

File: USPT

Dec 16, 1997

US-PAT-NO: 5698836

DOCUMENT-IDENTIFIER: US 5698836 A

TITLE: IC card

Full	Title	Citation	Front	Review	Classification	Date	Reference	Claims	KWIC	Draw. De
------	-------	----------	-------	--------	----------------	------	-----------	--------	------	----------

☐ 9. Document ID: US 5506396 A

L3: Entry 9 of 11

File: USPT

Apr 9, 1996

US-PAT-NO: 5506396

DOCUMENT-IDENTIFIER: US 5506396 A

TITLE: Microcomputer for IC card

Full	Title	Citation	Front	Review	Classification	Date	Reference	Claims	KWIC	Draw. De
------	-------	----------	-------	--------	----------------	------	-----------	--------	------	----------

☐ 10. Document ID: JP 2003160231 A

L3: Entry 10 of 11

File: JPAB

Jun 3, 2003

PUB-NO: JP02003160231A

DOCUMENT-IDENTIFIER: JP 2003160231 A
TITLE: ARTICLE DELIVERY ADDRESS SPECIFYING METHOD

Full	Title	Citation	Front	Review	Classification	Date	Reference			Claims	KWC	Draw. De
------	-------	----------	-------	--------	----------------	------	-----------	--	--	--------	-----	----------

Clear	Generate Collection	Print	Fwd Refs	Bkwd Refs	Generate OACS
-------	---------------------	-------	----------	-----------	---------------

Terms	Documents
L2 and collat\$	11

Display Format:

[Previous Page](#)

[Next Page](#)

[Go to Doc#](#)

Hit List

Clear	Generate Collection	Print	Fwd Refs	Bkwd Refs
Generate OACS				

Search Results - Record(s) 11 through 11 of 11 returned.

☐ 11. Document ID: JP 04267477 A

L3: Entry 11 of 11

File: JPAB

Sep 24, 1992

PUB-NO: JP404267477A

DOCUMENT-IDENTIFIER: JP 04267477 A

TITLE: RANGE RETRIEVING SYSTEM BY MESH CODING OF ADDRESS

Full	Title	Citation	Front	Review	Classification	Date	Reference	Abstract	Claims	KWIC	Draw. Des
------	-------	----------	-------	--------	----------------	------	-----------	----------	--------	------	-----------

Clear	Generate Collection	Print	Fwd Refs	Bkwd Refs	Generate OACS
-------	---------------------	-------	----------	-----------	---------------

Terms	Documents
L2 and collat\$	11

Display Format:

[Previous Page](#)

[Next Page](#)

[Go to Doc#](#)

[First Hit](#)

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)



Generate Collection

Print

L4: Entry 1 of 3

File: PGPB

Feb 28, 2002

PGPUB-DOCUMENT-NUMBER: 20020026281

PGPUB-FILING-TYPE: new

DOCUMENT-IDENTIFIER: US 20020026281 A1

TITLE: On-vehicle vehicle guide apparatus, communication server system, and substitute vehicle guide system

PUBLICATION-DATE: February 28, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Shibata, Makoto /	Saga		JP	
Moribe, Nobuyuki	Fukuoka		JP	
Yamaguchi, Tomonari	Fukuoka		JP	

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	COUNTRY	TYPE CODE
Fujitsu Limited	Kawasaki		JP	03

APPL-NO: 09/ 875920 [PALM]

DATE FILED: June 8, 2001

FOREIGN-APPL-PRIORITY-DATA:

COUNTRY	APPL-NO	DOC-ID	APPL-DATE
JP	PCT/JP99/06836	1999JP-PCT/JP99/06836	December 7, 1999
JP	10-350059	1998JP-10-350059	December 9, 1998

INT-CL: [07] G01 C 21/32

US-CL-PUBLISHED: 701/208; 340/995

US-CL-CURRENT: 701/208; 340/995.1

REPRESENTATIVE-FIGURES: 1

ABSTRACT:

There is provided an alternative vehicle guiding system wherein a communication sever system receives a fault condition information from vehicles and also transmits the necessary information to the navigation system of an alternative vehicle in order to prevent the damage of loads by quickly guiding the alternative vehicle to the fault generating vehicle position and then realizing smooth transfer of remaining work to the alternative vehicle.

The navigation system, fault monitoring means and vehicle communication means are respectively provided in the vehicles for transportation work and the center server

system is also provided to transmit the vehicle distribution information to the vehicles. When a fault is generated in the vehicle, the communication server system receives the information from this vehicle and the navigation system of the alternative vehicle displays the transmitted information. Thereby, the alternative vehicle can always be distributed smoothly without any manual operation, the time required until the alternative vehicle arrives and the work transfer time can be shortened and adverse effect on the loads and successive load distribution work can be prevented.

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)



DOCUMENT-IDENTIFIER: US 20020026281 A1

TITLE: On-vehicle vehicle guide apparatus, communication server system, and substitute vehicle guide system

Application Filing Date:
20010608Summary of Invention Paragraph:

[0005] In this navigation system, the current position information of the vehicle mounting this system is obtained mainly using GPS (Global Positioning System) as the measuring means to measure the current position of the vehicle, this position information is collated with the map information stored in a storage device as the map database within the vehicle and the map of the predetermined range in the periphery of the current position is displayed on the display area and the current position of the vehicle is displayed on this map. When the position information of the destination (latitude and longitude) is inputted as required from an operation terminal, this position information of the destination is collated with the map database and the route information up to the destination is displayed on the map. When a driver previously inputs the position information of the destination with hands or voices, the current position is detected with the navigation system while the vehicle is running and the map in which the route to the destination is indicated is displayed on the display area in view of guiding the vehicle to the destination.

Summary of Invention Paragraph:

[0011] In order to attain the object explained above, the mobile-type vehicle guiding apparatus of the present invention comprises a navigation system comprising a measuring means for obtaining the current position, a map database storing the map information and a display area that can display characters and video information and assures visual recognition of a driver for displaying the current position of the vehicle on the display area together with the map of peripheral areas by collating the current position information obtained from the measuring means with the map information of the map database, a fault monitoring means for notifying a fault condition information to a driver with an audio output and/or the predetermined display that may be visually recognized with a driver by detecting the fault condition in the loading room of the vehicle and a vehicle communication means for receiving information transmitted from external side through the radio communication link and also transmitting, to the external side, the information obtained in the vehicle side including the current position information of the vehicle obtained with the measuring means of the navigation system and the fault condition information of vehicle obtained with the fault monitoring means with the radio communication link at least when a fault condition is generated, whereby at least the destination position information received with the vehicle communication means is collated with the map information of the map database and the route information up to the destination from the current position of vehicle is displayed on the display area.

Summary of Invention Paragraph:

[0015] Moreover, in the alternative vehicle guiding system of the present invention, the communication server system explained above receives the information

acquired in the vehicle side including the fault condition information from the vehicle in which a fault is generated in the loading room through the radio communication link and transmits the fault generating vehicle position information and position information of relaying point and destination assigned to the fault generating vehicle and at which the fault generating vehicle does not arrive to the vehicle communication means of the other predetermined vehicle through the radio communication link and the navigation system of the predetermined vehicle explained above acquires the position information of relaying point and destination from the communication server system, collates the relevant position information with the map information of map database and displays the route information up to the current position of vehicle on the display area.

Detail Description Paragraph:

[0032] In each figure, the alternative vehicle guiding system in relation to this embodiment has a structure, comprising a navigation system 10 that is respectively mounted to a plurality of vehicles 60 for transportation use, with inclusion of a measuring means 10a for measuring the current position, a map database 10b for storing a map information and a display area 10c that can display the character and video information for visual recognition of a driver in order to display the current position of a vehicle 60 on the display area 10c together with the map of the peripheral region by collating the current position information obtained from the measuring means 10a with the map information of the map database 10b, a fault monitoring means 20 mounted in each vehicle 60 to notify the fault condition information to a driver with the audio output or predetermined display which may be visually recognized with a driver by detecting a fault condition in the loading room, a mobile telephone 30 as the vehicle communication means mounted in each vehicle 60 to transmit the information obtained in the vehicle side to the external side through the mobile telephone network as the radio communication link and to also receive the information transmitted from the external side through the mobile telephone network, and a communication server system 40 for suddenly receiving, while detecting the running conditions of a vehicle 60 by transmitting, as the initial values, the position information of the relaying point and destination of transportation based on the work request respectively assigned previously to each vehicle 60 to the mobile telephone 30 of each vehicle 60 through the mobile telephone network and also always receiving the information obtained in the vehicle side from the mobile telephone 30 through the mobile telephone network, a fault condition information from any vehicle 60 and also transmitting, if it is determined that the relevant vehicle 60 cannot eliminate the fault condition, a fault generating vehicle position and a position information of the relaying point and destination at which the fault generating vehicle does not yet arrive that is assigned to the fault generating vehicle as the additional relaying point and/or destination to the mobile telephone 30 of the other predetermined vehicle 60 through the mobile telephone network.

Detail Description Paragraph:

[0033] The navigation system 10 acquires, as in the case of the related art explained above, the current position information of a vehicle 60 using GSP as the measuring means 10a for measuring the current position of the vehicle 60, collates this current position information with the map information of the map database 10b and displays, on the display area 10c, the map of the current position of vehicle 60 and the predetermined range of the peripheral area of the current position and moreover, as the new functions automatically extract various vehicle distribution instruction data including the position information (latitude and longitude) of the destination and relaying point transmitted from the communication server system 40, then collates the position information of the destination and relaying point with the map information and then displays the route up to the destination on the map of the display area 10c in order to guide the vehicle. The information transfer function to a driver from this navigation system 10 is composed of the map information display function to display, as the character information, the information such as address, telephone number and route distance or the like of the

destination and relaying point on the display area 10c and to print such information on a sheet of paper with a printer (not illustrated) provided in the vehicle and a route display function for automatically displaying the route up to the destination from the current position on the map of the display area 10c without any operation of a driver after the character information. Here, for the navigation system 10, a driver can input the position information of destination and relaying point manually or with hands as in the case of the related art.

Detail Description Paragraph:

[0043] If the fault condition of vehicle 60 cannot be recovered, the communication server system 40 acquires, first, the vehicle distribution information instructed to the fault generating vehicle 60, namely the information such as position information of load collecting or distributing points, load information like kind and amount of load and restricting condition for loading or the like as the information to instruct the vehicle distribution to the alternative vehicle (step 108), collates such information with the previously registered intrinsic information of each vehicle other than the fault generating vehicle and the information such as loading condition of each vehicle in this timing which can be determined from the initial vehicle distribution information and selects the prospective vehicles satisfying the conditions of the vehicle distribution information of the fault generating vehicle 60 (step 109).

Detail Description Paragraph:

[0048] In the vehicle 60 as the alternative vehicle, the navigation system 10 receives and acquires the emergency vehicle distribution instruction data via the mobile telephone 30 (step 116). The navigation system 10 displays the information such as the current position of the fault generating vehicle and route distance up to this position among the automatically fetched emergency vehicle distribution instruction data on the display area 10c as the character information (refer to FIG. 9(A)) (step 117), thereafter collates the current position information of the vehicle 60 obtained using the measuring means 10a with the map information of the map database 10b, displays the map of the current position of the vehicle 60 and predetermined range of the periphery of the current position on the display area 10c and simultaneously collates the current position information of the fault generating vehicle in the emergency vehicle distribution instruction data with the map information to display the route up to the fault generating vehicle overlapping on the map of the display area 10c (step 118).

Detail Description Paragraph:

[0050] The navigation system 10 of the alternative vehicle displays first the information such as address, telephone number and route distance or the like of the relaying points and destinations for transportation work as the character information (refer to FIG. 10(A)) on the display area 10c from the already fetched emergency vehicle distribution instruction data (step 119), thereafter collates the current position information of vehicle 60 obtained using the measuring means 10a with the map information of the map database 10b, displays the map of the current position of vehicle 60 and predetermined range in the periphery of the current position on the display area 10c, collates the position information of the destinations and relaying points in the emergency vehicle distribution instruction data with the map information and displays the route up to the destinations overlapping on the map of the display area 10c (step 120).

Detail Description Paragraph:

[0054] Moreover, in the alternative vehicle guiding system in relation to this embodiment, as the information transfer function of the navigation system 10, the information such as fault generating vehicle position, destination and relaying point, telephone number and route distance or the like is displayed on the display area 10c as the character information and the route from the current position is displayed on the map of the display area 10c, but moreover it is also possible, as the audio information output function, that the information to be displayed as the

character information is simultaneously read and outputted as the audio information and the route guidance from the current position is executed by voicing the audio output. Accordingly, understanding of driver can further be enhanced, frequency to watch the display area 10c can be reduced and risk of drive can also be reduced. In addition, it is also possible to introduce the structure that the display area 10c of the navigation system 10 is divided and operation procedures, current display contents and other information such as emergency message can be displayed with characters or images on the non-displaying display area in parallel with the display of the character information and map information. Therefore, it is no longer required for the other displays to switch the display area of the character information and map information and thereby each information can surely be transferred without giving any bewildering to the driver watching the display area 10c.

CLAIMS:

1. A mobile vehicle guiding apparatus, comprising: navigation system including a measuring means for obtaining the current position, a map database for storing map information and display area for displaying characters and video information for visual recognition of a driver to collate the current position information acquired from said measuring means with the map information of said map database in view of displaying the current position of vehicle on said display area together with the map of the peripheral area of the current position; fault monitoring means to detect a fault condition in the loading room of vehicle in view of notifying the fault condition information to a driver with the voice output and/or with the predetermined display that may be visually recognized with a driver; and vehicle communication means to receive the information transmitted from the external side through the radio communication link and to transmit, at least when a fault condition is generated, the information acquired in the vehicle side including the current position information of vehicle obtained with the measuring means of said navigation system and vehicle fault condition information obtained with said fault monitoring means to the external side through the radio communication link; whereby the position information of at least destination received with said vehicle communication means is collated with the map information of said map database and the route information up to the destination from the current position of vehicle can be displayed on said display area.

3. An alternative vehicle guiding system, wherein the mobile vehicle guiding apparatus as claimed in claim 1 is provided in a plurality of vehicles for transportation work, the communication server system as claimed in claim 2 receives the information acquired in the vehicle side including the fault condition information transmitted from the vehicle generating a fault condition in its loading room via said radio communication link and transmits the fault generating vehicle position and the position information, assigned to the fault generating vehicle, of the relaying point and destination at which the fault generating vehicle does not arrive to the vehicle communication means of the other predetermined vehicle via the radio communication link, and the navigation system of said predetermined vehicle acquires the position information of relaying point and destination from said communication server system received with said vehicle communication means, collates the relevant position information with the map information of said map database and displays the route information up to the destination from the current position of vehicle on said display area.

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)

[First Hit](#) [Previous Doc](#) [Next Doc](#) [Go to Doc#](#)

End of Result Set

☐ [Generate Collection](#) [Print](#)

L3: Entry 11 of 11

File: JPAB

Sep 24, 1992

PUB-NO: JP404267477A

DOCUMENT-IDENTIFIER: JP 04267477 A

TITLE: RANGE RETRIEVING SYSTEM BY MESH CODING OF ADDRESS

PUBN-DATE: September 24, 1992

INVENTOR-INFORMATION:

NAME

COUNTRY

HORI, TAKAYUKI

ASSIGNEE-INFORMATION:

NAME

COUNTRY

KYUSHU NIPPON DENKI SOFTWARE KK

APPL-NO: JP03028304

APPL-DATE: February 22, 1991

INT-CL (IPC): G06F 15/40; G06F 15/40; G06F 15/40

ABSTRACT:

PURPOSE: To facilitate the retrieve processing of address data in an arbitrary range with the quantity of data and to prevent the retrieving processing and the address data from being affected even when the names of cities, towns and villages are changed, with the geographical two-dimensional data by mesh- coding the address data on a map.

CONSTITUTION: All retrieval object areas on the map are divided into equal intervals in an X-Y axis direction and a code is imparted (steps 101 and 102.) The X-axis and Y-axis codes are imparted to each address present in the framework on the map shown by the X-axis and Y--axis codes, the address is mesh- coded and stored(step 103). By the X-axis and Y-axis codes on the map, the retrieval range is designated with an FROM.TO method(step 201). The mesh code in the range shown with the FROM.TO code designated is generated, the mesh code in the range and the mesh code of the stored address are collated with each other and the coincident and corresponding address data are outputted as the retrieved result (steps 202 and 203).

COPYRIGHT: (C)1992, JPO&Japio

[Previous Doc](#) [Next Doc](#) [Go to Doc#](#)

[First Hit](#) [Previous Doc](#) [Next Doc](#) [Go to Doc#](#)

End of Result Set



Generate Collection

Print

L3: Entry 11 of 11

File: JPAB

Sep 24, 1992

DOCUMENT-IDENTIFIER: JP 04267477 A

TITLE: RANGE RETRIEVING SYSTEM BY MESH CODING OF ADDRESS

Abstract Text (2):

CONSTITUTION: All retrieval object areas on the map are divided into equal intervals in an X-Y axis direction and a code is imparted (steps 101 and 102.) The X-axis and Y-axis codes are imparted to each address present in the framework on the map shown by the X-axis and Y--axis codes, the address is mesh- coded and stored(step 103). By the X-axis and Y-axis codes on the map, the retrieval range is designated with an FROM.TO method(step 201). The mesh code in the range shown with the FROM.TO code designated is generated, the mesh code in the range and the mesh code of the stored address are collated with each other and the coincident and corresponding address data are outputted as the retrieved result (steps 202 and 203).

Application Date (1):

19910222

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)



Generate Collection

Print

L4: Entry 2 of 3

File: USPT

Nov 19, 2002

DOCUMENT-IDENTIFIER: US 6484091 B2

TITLE: On-vehicle vehicle guide apparatus, communication server system, and substitute vehicle guide system

Application Filing Date (1):
20010608Brief Summary Text (6):

In this navigation system, the current position information of the vehicle mounting this system is obtained mainly using GPS (Global Positioning System) as the measuring means to measure the current position of the vehicle, this position information is collated with the map information stored in a storage device as the map database within the vehicle and the map of the predetermined range in the periphery of the current position is displayed on the display area and the current position of the vehicle is displayed on this map. When the position information of the destination (latitude and longitude) is inputted as required from an operation terminal, this position information of the destination is collated with the map database and the route information up to the destination is displayed on the map. When a driver previously inputs the position information of the destination with hands or voices, the current position is detected with the navigation system while the vehicle is running and the map in which the route to the destination is indicated is displayed on the display area in view of guiding the vehicle to the destination.

Brief Summary Text (13):

In order to attain the object explained above, the mobile-type vehicle guiding apparatus of the present invention comprises a navigation system comprising a measuring means for obtaining the current position, a map database storing the map information and a display area that can display characters and video information and assures visual recognition of a driver for displaying the current position of the vehicle on the display area together with the map of peripheral areas by collating the current position information obtained from the measuring means with the map information of the map database, a fault monitoring means for notifying a fault condition information to a driver with an audio output and/or the predetermined display that may be visually recognized with a driver by detecting the fault condition in the loading room of the vehicle and a vehicle communication means for receiving information transmitted from external side through the radio communication link and also transmitting, to the external side, the information obtained in the vehicle side including the current position information of the vehicle obtained with the measuring means of the navigation system and the fault condition information of vehicle obtained with the fault monitoring means with the radio communication link at least when a fault condition is generated, whereby at least the destination position information received with the vehicle communication means is collated with the map information of the map database and the route information up to the destination from the current position of vehicle is displayed on the display area.

Brief Summary Text (17):

Moreover, in the alternative vehicle guiding system of the present invention, the communication server system explained above receives the information acquired in

the vehicle side including the fault condition information from the vehicle in which a fault is generated in the loading room through the radio communication link and transmits the fault generating vehicle position information and position information of relaying point and destination assigned to the fault generating vehicle and at which the fault generating vehicle does not arrive to the vehicle communication means of the other predetermined vehicle through the radio communication link and the navigation system of the predetermined vehicle explained above acquires the position information of relaying point and destination from the communication server system, collates the relevant position information with the map information of map database and displays the route information up to the current position of vehicle on the display area.

Detailed Description Text (3):

In each figure, the alternative vehicle guiding system in relation to this embodiment has a structure, comprising a navigation system 10 that is respectively mounted to a plurality of vehicles 60 for transportation use, with inclusion of a measuring means 10a for measuring the current position, a map database 10b for storing a map information and a display area 10c that can display the character and video information for visual recognition of a driver in order to display the current position of a vehicle 60 on the display area 10c together with the map of the peripheral region by collating the current position information obtained from the measuring means 10a with the map information of the map database 10b, a fault monitoring means 20 mounted in each vehicle 60 to notify the fault condition information to a driver with the audio output or predetermined display which may be visually recognized with a driver by detecting a fault condition in the loading room, a mobile telephone 30 as the vehicle communication means mounted in each vehicle 60 to transmit the information obtained in the vehicle side to the external side through the mobile telephone network as the radio communication link and to also receive the information transmitted from the external side through the mobile telephone network, and a communication server system 40 for suddenly receiving, while detecting the running conditions of a vehicle 60 by transmitting, as the initial values, the position information of the relaying point and destination of transportation based on the work request respectively assigned previously to each vehicle 60 to the mobile telephone 30 of each vehicle 60 through the mobile telephone network and also always receiving the information obtained in the vehicle side from the mobile telephone 30 through the mobile telephone network, a fault condition information from any vehicle 60 and also transmitting, if it is determined that the relevant vehicle 60 cannot eliminate the fault condition, a fault generating vehicle position and a position information of the relaying point and destination at which the fault generating vehicle does not yet arrive that is assigned to the fault generating vehicle as the additional relaying point and/or destination to the mobile telephone 30 of the other predetermined vehicle 60 through the mobile telephone network.

Detailed Description Text (4):

The navigation system 10 acquires, as in the case of the related art explained above, the current position information of a vehicle 60 using GSP as the measuring means 10a for measuring the current position of the vehicle 60, collates this current position information with the map information of the map database 10b and displays, on the display area 10c, the map of the current position of vehicle 60 and the predetermined range of the peripheral area of the current position and moreover, as the new functions automatically extract various vehicle distribution instruction data including the position information (latitude and longitude) of the destination and relaying point transmitted from the communication server system 40, then collates the position information of the destination and relaying point with the map information and then displays the route up to the destination on the map of the display area 10c in order to guide the vehicle. The information transfer function to a driver from this navigation system 10 is composed of the map information display function to display, as the character information, the information such as address, telephone number and route distance or the like of the

destination and relaying point on the display area 10c and to print such information on a sheet of paper with a printer (not illustrated) provided in the vehicle and a route display function for automatically displaying the route up to the destination from the current position on the map of the display area 10c without any operation of a driver after the character information. Here, for the navigation system 10, a driver can input the position information of destination and relaying point manually or with hands as in the case of the related art.

Detailed Description Text (14):

If the fault condition of vehicle 60 cannot be recovered, the communication server system 40 acquires, first, the vehicle distribution information instructed to the fault generating vehicle 60, namely the information such as position information of load collecting or distributing points, load information like kind and amount of load and restricting condition for loading or the like as the information to instruct the vehicle distribution to the alternative vehicle (step 108), collates such information with the previously registered intrinsic information of each vehicle other than the fault generating vehicle and the information such as loading condition of each vehicle in this timing which can be determined from the initial vehicle distribution information and selects the prospective vehicles satisfying the conditions of the vehicle distribution information of the fault generating vehicle 60 (step 109).

Detailed Description Text (19):

In the vehicle 60 as the alternative vehicle, the navigation system 10 receives and acquires the emergency vehicle distribution instruction data via the mobile telephone 30 (step 116). The navigation system 10 displays the information such as the current position of the fault generating vehicle and route distance up to this position among the automatically fetched emergency vehicle distribution instruction data on the display area 10c as the character information (refer to FIG. 9(A)) (step 117), thereafter collates the current position information of the vehicle 60 obtained using the measuring means 10a with the map information of the map database 10b, displays the map of the current position of the vehicle 60 and predetermined range of the periphery of the current position on the display area 10c and simultaneously collates the current position information of the fault generating vehicle in the emergency vehicle distribution instruction data with the map information to display the route up to the fault generating vehicle overlapping on the map of the display area 10c (step 118).

Detailed Description Text (21):

The navigation system 10 of the alternative vehicle displays first the information such as address, telephone number and route distance or the like of the relaying points and destinations for transportation work as the character information (refer to FIG. 10(A)) on the display area 10c from the already fetched emergency vehicle distribution instruction data (step 119), thereafter collates the current position information of vehicle 60 obtained using the measuring means 10a with the map information of the map database 10b, displays the map of the current position of vehicle 60 and predetermined range in the periphery of the current position on the display area 10c, collates the position information of the destinations and relaying points in the emergency vehicle distribution instruction data with the map information and displays the route up to the destinations overlapping on the map of the display area 10c (step 120).

Detailed Description Text (25):

Moreover, in the alternative vehicle guiding system in relation to this embodiment, as the information transfer function of the navigation system 10, the information such as fault generating vehicle position, destination and relaying point, telephone number and route distance or the like is displayed on the display area 10c as the character information and the route from the current position is displayed on the map of the display area 10c, but moreover it is also possible, as the audio information output function, that the information to be displayed as the

character information is simultaneously read and outputted as the audio information and the route guidance from the current position is executed by voicing the audio output. Accordingly, understanding of driver can further be enhanced, frequency to watch the display area 10c can be reduced and risk of drive can also be reduced. In addition, it is also possible to introduce the structure that the display area 10c of the navigation system 10 is divided and operation procedures, current display contents and other information such as emergency message can be displayed with characters or images on the non-displaying display area in parallel with the display of the character information and map information. Therefore, it is no longer required for the other displays to switch the display area of the character information and map information and thereby each information can surely be transferred without giving any bewildering to the driver watching the display area 10c.

CLAIMS:

1. A mobile vehicle guiding apparatus, comprising: a navigation system including measuring means for obtaining a current vehicle position information, a map database storing map information and a display displaying characters and video information for visual recognition by a driver of the vehicle to collate the current vehicle position information obtained by said measuring means with the map information of said map database thereby to display the current position of the vehicle on said display together with a map of a peripheral area relative to the current vehicle position; fault monitoring means for detecting information of a fault condition in the vehicle and notifying the fault condition information to the driver with an audio message and/or by a predetermined display that is visually recognizable by the driver; and vehicle communication means for receiving information transmitted to the vehicle from an external side through a radio communication link and for transmitting, at least when a fault condition is detected, the information detected in the vehicle side, including current vehicle position information obtained by the measuring means of said navigation system and said vehicle fault condition information detected by said fault monitoring means, to the external side through the radio communication link, wherein the position information of at least a vehicle destination received by said vehicle communication means is collated with the map information of said map database and route information to the vehicle destination from the current vehicle position is displayed on said display area.

3. An alternative vehicle guiding system comprising: in each of a plurality of vehicles for transportation work, a mobile vehicle guiding apparatus, comprising: a navigation system including measuring means for obtaining a current vehicle position information, a map database storing map information and a display to display characters and video information for visual recognition by a driver of the vehicle to collate the current vehicle position information obtained by said measuring means with the map information of said map database thereby to display the current position of the vehicle on said display together with a map of a peripheral area relative to the current vehicle position; fault monitoring means for detecting information of a fault condition in the vehicle and notifying the fault condition information to the driver with an audio message and/or by a predetermined display that is visually recognizable by the driver; and vehicle communication means for receiving information transmitted to the vehicle from an external side through a radio communication link and to transmit, at least when a fault condition is detected, the information detected in the vehicle side, including current vehicle position information obtained by the measuring means of said navigation system and said vehicle fault condition information detected by said fault monitoring means, to the external side through the radio communication link, wherein the position information of at least a vehicle destination received by said vehicle communication means is collated with the map information of said map database and route information to the vehicle destination from the current vehicle position is displayed on said display area; and a communication server

system issuing work instructions by transmitting a position information of relaying point and destination for transportation work to the vehicle communication means of each vehicle through the radio communication link among a plurality of vehicles for transportation work based on a preset vehicle distribution information, wherein the position information of relaying point and destination for transportation work based on the vehicle distribution information is transmitted as the initial value to a plurality of vehicles, thereafter the information acquired in the vehicle side including a fault condition information from a certain vehicle is suddenly received, if it is determined that said vehicle cannot eliminate the fault condition, a relevant fault generating vehicle position and the relaying point and destination at which the fault generating vehicle does not yet arrive assigned to the fault generating vehicle are additionally assigned as the relaying point and/or destination of another predetermined vehicle and the added position information of relaying point and/or destination is transmitted to the vehicle communication means of said predetermined vehicle via the radio communication link, wherein the communication server system receives the information acquired in the vehicle side including the fault condition information transmitted from the vehicle generating a fault condition in its loading room via said radio communication link and transmits the fault generating vehicle position and the position information, assigned to the fault generating vehicle, of the relaying point and destination at which the fault generating vehicle does not arrive to the vehicle communication means of the other predetermined vehicle via the radio communication link, and the navigation system of said predetermined vehicle acquires the position information of relaying point and destination from said communication server system received with said vehicle communication means, collates the relevant position information with the map information of said map database and displays the route information up to the destination from the current position of vehicle on said display area.

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)

[First Hit](#) [Fwd Refs](#)

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)

End of Result Set



Generate Collection

Print

L4: Entry 3 of 3

File: USPT

Sep 7, 1999

US-PAT-NO: 5948040

DOCUMENT-IDENTIFIER: US 5948040 A

TITLE: Travel reservation information and planning system

DATE-ISSUED: September 7, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
DeLorme; David M.	Yarmouth	ME		
Gray; Keith A.	Dresden	ME		
Ferguson; T. Angus	Portland	ME		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
DeLorme Publishing Co.	Yarmouth	ME			02

APPL-NO: 08/ 797471 [PALM]

DATE FILED: February 6, 1997

PARENT-CASE:

CROSS-REFERENCE TO RELATED APPLICATIONS This patent application is a continuation-in-part (CIP) of the David M. DeLorme et al. U.S. patent application Ser. No. 08/661,600 filed Jun. 11, 1996, for COMPUTER AIDED ROUTING AND POSITIONING SYSTEM, now U.S. Pat. No. 5,802,492 which is a CIP of the David M. DeLorme et al. U.S. patent application Ser. No. 08/381,214 filed Jan. 31, 1995 for COMPUTER AIDED ROUTING SYSTEM, now U.S. Pat. No. 5,559,707, issued Sep. 24, 1996, which is a CIP of the David M. DeLorme et al. U.S. patent application Ser. No. 08/265,327 filed Jun. 24, 1994 for COMPUTER AIDED MAP LOCATION SYSTEM now abandoned. This patent application is also a CIP of the Keith A. Gray U.S. patent application Ser. No. 08/521,828 filed on Aug. 31, 1995, for COMPUTERIZED ADDRESS LOCATION AND COMMUNICATION SYSTEM now abandoned. All of the cross-referenced applications have a common assignee who is the assignee of the present application. The contents of these related patent applications are incorporated herein by reference.

INT-CL: [06] G06 F 19/00, G01 C 21/00

US-CL-ISSUED: 701/201; 701/208, 701/211, 340/990, 705/5

US-CL-CURRENT: 701/201; 340/990, 701/208, 701/211, 705/5

FIELD-OF-SEARCH: 701/201, 701/202, 701/207, 701/208, 701/209, 701/211, 701/212, 701/213, 705/5, 705/6, 340/988, 340/989, 340/990, 340/995

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected

Search ALL

Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<input type="checkbox"/>	<u>4359631</u>	November 1982	Lockwood et al.	360/12
<input type="checkbox"/>	<u>4862357</u>	August 1989	Ahlstrom et al.	705/6
<input type="checkbox"/>	<u>4926336</u>	May 1990	Yamada	364/444
<input type="checkbox"/>	<u>5021953</u>	June 1991	Webber et al.	705/6
<input type="checkbox"/>	<u>5172321</u>	December 1992	Ghaem et al.	701/202
<input type="checkbox"/>	<u>5191523</u>	March 1993	Whitesage	705/6
<input type="checkbox"/>	<u>5208756</u>	May 1993	Song	364/449
<input type="checkbox"/>	<u>5231584</u>	July 1993	Nimura et al.	364/444
<input type="checkbox"/>	<u>5237499</u>	August 1993	Garback	705/5
<input type="checkbox"/>	<u>5243528</u>	September 1993	Lefebvre	701/211
<input type="checkbox"/>	<u>5253166</u>	October 1993	Dettebach et al.	705/5
<input type="checkbox"/>	<u>5272638</u>	December 1993	Martin et al.	701/202
<input type="checkbox"/>	<u>5331546</u>	July 1994	Webber et al.	705/6
<input type="checkbox"/>	<u>5353034</u>	October 1994	Sato et al.	340/988
<input type="checkbox"/>	<u>5359527</u>	October 1994	Takanabe et al.	364/449
<input type="checkbox"/>	<u>5369588</u>	November 1994	Hayami et al.	701/209
<input type="checkbox"/>	<u>5422809</u>	June 1995	Griffin et al.	705/5
<input type="checkbox"/>	<u>5444618</u>	August 1995	Seki et al.	364/420
<input type="checkbox"/>	<u>5519619</u>	May 1996	Seda	701/201
<input type="checkbox"/>	<u>5537324</u>	July 1996	Nimura et al.	364/449
<input type="checkbox"/>	<u>5587911</u>	December 1996	Asano et al.	364/444.2
<input type="checkbox"/>	<u>5724520</u>	March 1998	Goheen	705/5

OTHER PUBLICATIONS

Makulowich, John, "Traveling by Virtual Reservation," Washington Technology, Jan. 23, 1997, p. 42.
 Knecht, Bruce, G., "Microsoft Puts Newspapers in Highanxiety.com," The Wall Street Journal, Jul. 15, 1996, pp. B1, B10.
 "InforTravel Expands Service," Business Geographics, vol. 4, No. 6, Jun., 1996, p. 13.
 DelRosso, Laura, "Firm Customizes Internet Res Link," Travel Weekly, vol. 55, No. 26, Apr. 1, 1996, pp. 43-44, 47.
 "Casto Travel's Resource Library," www.casto.com.
 "Sunnyside Computing, Inc.," www.itn.net.

ART-UNIT: 361

PRIMARY-EXAMINER: Nguyen; Tan

ABSTRACT:

Computerized travel reservation information and planning system that generates "map ticket" output in various media, for guidance and transactions en route. Such print or electronic documents can include bar or alphanumeric codes for automated recognition and/or access. WHERE?, WHO/WHAT?, WHEN? and HOW? menus enable flexible user inquiries accessing selectable geographic, topical, temporal and transactional data records and relational processing. Sub-menus provide further capabilities: e.g. routing, topical searching; searches of events calendars, almanacs, appointment books, related itinerary scheduling; trip budgeting issues, plus travel arrangement availabilities or other goods/services offers. Online communications links access updated or supplemental information on places, times, topics and other provider goods/service offers. Online computer-aided routing system enables input of selectable travel origin, destination, and waypoints to compute travel routes, available transportation services, costs, options, and schedules. A point-of-interest database lets users pick types of attractions or accommodations within a user-selected region around routes of travel. Users engage in an iterative planning process, revising or editing travel plans, previewing travelogs of alternate routes, selecting point of interest parameters, comparing times and costs of transportation options, in order to achieve a satisfactory travel plan. The system provides printed or electronic output that may include any one or more of text itinerary, ordered set of travel maps, customized collection of information on points of interest information and a selected array of valid reservation confirmations, tickets and/or discount coupons coded with elements for automated recognition and processing. Mobile users, including GPS-linked users, can access the system via wireless communication units.

80 Claims, 16 Drawing figures

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)

End of Result Set



Generate Collection

Print

L4: Entry 3 of 3

File: USPT

Sep 7, 1999

DOCUMENT-IDENTIFIER: US 5948040 A

TITLE: Travel reservation information and planning system

Application Filing Date (1):19970206Brief Summary Text (36):

In the preferred example, the TRIPS software is composed of a reservation-information-and-planning system linked to one or more travel service provider. The TRIPS user can be provided with communications links for online communication and transfer of reservation data, ticketing data, spatially related data, and software tools for map reading between computers and between users. For example a TRIPS user may communicate with another TRIPS system or user for transfer of user location data and any other spatially related data. In addition to a travel service providing reservation and ticketing data, the TRIPS user can also communicate with external databases, a central communications service bureau, and on-line mapping services for latest information relating to loc/objects, routes, and map modifications, priority messages, etc.

Detailed Description Text (39):

As shown in FIG. 1C at 152, a preferred TRIPS embodiment defaults to a graphical user interface (GUI) in the form of a dynamic, multi-scale map display readily manipulated and queried by the user. Such map display GUIs are described in issued patents and pending patent applications assigned to DeLorme Publishing Co., Inc., also owner of this TRIPS patent disclosure. See e.g.: David M. DeLorme, U.S. Pat. No. 4,972,319 "ELECTRONIC GLOBAL MAP GENERATING SYSTEM"; David M. DeLorme, U.S. Pat. No. 5,030,117 "DIGITAL GLOBAL MAP GENERATING SYSTEM"; David M. DeLorme & Keith A. Gray, U.S. patent application Ser. No. 08/265,327 "COMPUTER-AIDED MAP LOCATION SYSTEM" [or CAMLS] filed Jun. 24, 1994; David M. DeLorme & Keith A. Gray, U.S. Pat. No. 5,559,707 "COMPUTER-AIDED ROUTING SYSTEM" [or CARS]; Keith A. Gray U.S. patent application Ser. No. 08/521,828 "COMPUTERIZED ADDRESS LOCATION AND COMMUNICATION SYSTEM" [or CALCS] filed Aug. 31, 1995; David M. DeLorme & Keith A. Gray, U.S. patent application Ser. No. 08/661,600 "COMPUTER-AIDED ROUTING & POSITIONING SYSTEM" [or CARPS] filed Jun. 11, 1996. The preferred map display GUI at 152 in FIG. 1C embodies such capabilities e.g.: to zoom to different scale maps with variable resolution or levels of detail; to pan or shift seamlessly across to other map locations i.e. other latitudes and longitudes; and to locate on map displays named places, zip code and phone exchange areas, street addresses, or other landmarks and/or ordinary language geographic location and direction identifiers. Users can also get supplemental or updated information on points of interest or POIs, including multimedia previews on places near optimum computed travel routes. The map display technology in the patents and applications just mentioned facilitates the communication or electronic transfer of discrete, compact files or packets of map-related information between remote computers equipped with compatible mapping technology and/or to and from auxiliary devices like highly portable GPS receivers or other handheld digital travel aids. Digital displays of selectively detailed geographic information can be used in conjunction with sheet media printed maps with reference to named map grids for coordination and

correlation.

Detailed Description Text (73):

For yet another illustration of this general concept, suppose John Jones turns to the Accounting Subsystem at 227, in order to arrange for a convenient, hopefully not too expensive, round trip plane flight to his grandmother's birthday party--after entry or input of the date/time of the happy event in the Temporal Subsystem 223 in FIG. 2. This date/time input results in modifications to the TEMPORAL DATA sub-structure in FIG. 3. Also or instead, John Jones might turn to the Geographic Subsystem at 221, more specifically the ROUTES sub-menu within the WHERE? main menu for input at 155 in FIG. 1C, in order to consider optimum routes for the journey from his home in Knox, Ind. to and from the birthday party location. For example, John Jones might consult TRIPS about riding his motorcycle instead of flying, comparing costs and travel times. At any rate, whatever tasks or operations John Jones opts to do next in his TRIPS travel planning session can be facilitated--because, in anthropomorphic terms, the TRIPS software "knows" the date/time central to John Jones' travel plans. To enhance or automate subsequent operations, among other functions, this date/time information was memorized in the TEMPORAL DATA sub-structure, shown in FIG. 3, which is a standard part of the "data packet" underlying John Jones' individual travel planning session. Thus, TRIPS can take the time/date of the birthday party into account in any subsequent phases of John Jones' session in TRIPS including subsequent iterative operations in the Temporal Subsystem 223 itself (e.g. collateral scheduling issues and tasks). Selective sequencing and integration of the steps or operations involved in typical individual TRIPS travel planning sessions are further described hereinafter.

Detailed Description Text (77):

TRIPS operations commence at 401 in FIG. 4, with a typical "splash screen" or "home page" Greeting 403 which introduces the TRIPS Internet travel planning site's features, capabilities and rules. At 404, inveterate "surfers" can opt for instant access to the Main Menu at 341, and explore limited travel information. Alternatively, the user can choose the "SEE TERMS" prompt at 404, and study the Internet site Terms+Conditions at 405, including: e.g. legal notices, licensing and contractual terms, restrictions on copies, uses and liability; explanation of the user or member registration protocols, benefits and obligations; payment/credit terms; and the like. The opening "home page" also often displays other announcements or messages: e.g. trial offers, promoting new TRIPS site services and functions; advertising, typically placed by participating third-party providers of TRIPS travel information/services; instructions or tips for the user; incentives encouraging user registration or membership enrollment; and so forth.

Detailed Description Text (187):

FIG. 9 further illustrates facilities for managing output 925 within the WCU 907 including varied selection or combination of audio 927, text 929 and/or graphics 931. Such varied outputs express travel information and arrangement responses communicated from the TRIPS provider at 904 by return wireless communications 903 back to the user issuing a request at 907. These varied outputs also provide further selectivity or specificity in input of standard user requests. For example, after a primary input or "push-button" operation prompting a RESCUE 916 or a RESERVATIONS 920 request for example, the local WCU controller 912 or the TRIPS online provider 904 can respond with text 929, audio 927 and/or graphics 931 message or user selectable menu or set of options: e.g. "Car repair shop, Towing or Gas Station?"; "Medical Rescue, Police or Fire"; "Restaurant, Hotel or Campground"; and so forth--with a corresponding means for the remote user to select one or more specific options. Preferred embodiments for in-vehicle use provide audio 927 output of such selections, and even voice recognition technology for TRIPS user inputs and selection, for a user interface with minimal visual distraction for the vehicle driver. Text and graphics--for example, map displays--are preferred for vehicle passengers and other remote TRIPS retail users without visual distraction concerns. Travel information and arrangement responses from TRIPS 904 are also preferably

expressed by varied selection or combination of audio 927, text 929 and/or graphics 931 fitting the needs and circumstances of the remote TRIPS user. For example, audio 927 is preferred for travel directions output for vehicle drivers keeping their eyes on the road. Passengers are enabled to navigate with added text 929 and/or graphics 931 including map displays. The responsive output 925, 927, 929 and/or 931--as transmitted from the TRIPS provider 904 and expressed by the remote TRIPS user's WCU 907--can include citation of particular map grid names from a system of named map grids for use, coordination and/or correlation with corresponding printed maps 933 employed by the TRIPS user at the remote location.

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)

Carr, Jim
Network Magazine, 32
July 1, 2001

10/718670

ISSN: 1093-8001 LANGUAGE: English RECORD TYPE: Fulltext; Abstract
WORD COUNT: 936 LINE COUNT: 00079

...ABSTRACT: in the US. The service is based on a global network of 60 servers that maps IP addresses to geographic locations.

... moving to GeoPoint, Amazing Media employed five IT staffers to operate and manage a homegrown mapping facility. Deploying GeoPoint allowed the company to move these workers into "more strategic" positions, says...

...SERVERS

Quova's GeoPoint service is based on a global network of 60 servers that maps IP addresses to physical locations, according to Alexander. In the "commercial" part of the Web...

...and government sites and IP addresses inside corporate networks, the company has collected, analyzed, and mapped the IP address and geographic location information on about 1.4 billion IP addresses, she...

...Web surfer hits a GeoPoint-enabled site, that API communicates with Quova's DDS, which correlates the IP address into a geographical location and returns the information to the customer server, which can then

...

20010701

6/3,KWIC/3 (Item 1 from file: 256)
DIALOG(R)File 256:TecInfoSource
(c) 2005 Info.Sources Inc. All rts. reserv.

00121534 DOCUMENT TYPE: Review

PRODUCT NAMES: Go2.com (785172); MyAlladin.com (785181); local.info (785199)

TITLE: Wandering Web Maps To Real Locations
AUTHOR: Spangler, Todd
SOURCE: Interactive Week, v6 n46 p12(1) Nov 8, 1999
ISSN: 1078-7259

Homepage: <http://www.interactive-week.com>

RECORD TYPE: Review

REVIEW TYPE: Product Analysis

GRADE: Product Analysis, No Rating

REVISION DATE: 20030430

TITLE: Wandering Web Maps To Real Locations

...that will help users find specific locations, and its directory service's hierarchical name and address system will correlate to actual locations. NeoPoint's myAlladin.com compiles content that is location-based, from sites that include GetThere.com, InforSpace.com, and MapQuest .com, and then will deliver the information to mobile phones by the Wireless Access Protocol...

6/3,KWIC/4 (Item 1 from file: 674)

DIALOG(R)File 674:Computer News Fulltext

(c) 2005 IDG Communications. All rts. reserv.

083673

Freedom from IP address overload

Lucent's QIP Enterprise is the best tool for ending your IP addressing nightmare.

Byline: BARRY NANCE, NETWORK WORLD TEST ALLIANCE

Journal: Network World Page Number: 67

Publication Date: May 01, 2000

Word Count: 2978 Line Count: 281

Text:

... features make it the best DHCP and DNS tool.We found the user-to-address mapping of Check Point 's Meta IP worth mentioning as a great time saver. Network TeleSystems...has no self-imposed limits. Both editions offer what Check Point terms user-to-address mapping , a highly useful function that detects logons and equates logon account IDs with assigned IP...

... from a lower score for administration in our scorecard.Meta IP 's user-to-address mapping technology made our jobs as IP address administrators a snap. In contrast to specifying users ' MAC addresses or subnet IDs, user-to-address mapping is clearly a superior method of

identifying IP clients. Using Meta IP, we mapped dynamically-assigned IP addresses to clients based on logon account ID, logon time and MAC address . By automatically correlating file server logons with dynamic IP address leases, Meta IP let us painlessly track address assignments by user rather than by machine...

6/3,KWIC/5 (Item 1 from file: 63)
DIALOG(R)File 63:Transport Res(TRIS)
(c) fnt only 2005 Dialog Corp. All rts. reserv.

00200736 DA

TITLE: SOIL CONSERVATION SERVICE PERSONNEL

JOURNAL: Highway Research Board Bulletin

SUPPLEMENTAL NOTES: No 22-r, pp 101-109, 3 TAB

PUBLICATION DATE: 19570000 PUBLICATION YEAR: 1957

LANGUAGE: English SUBFILE: HRIS (H)

PUBLICATION DATE: 19570000

ABSTRACT: THE SOIL CONSERVATION SERVICE HAS PERSONNEL ENGAGE MAPPING , LABORATORY SOIL TESTING, INTERPRETATION, CLASSIFICAT COMPILATION, AND BASIC RESEARCH IN SOIL GENESIS, MORPHOLOGY GEOGRAPHY. THE MEMBERS OF THE STAFF...

...TECHNICAL PERSONNEL TO SUPERVISE THESE ACTIVITIES IN THE WASHINGTON-BELTSVILLE OFFICES ARE LISTED. NAMES AND ADDRES PRESENTED OF THE SOIL CORRELATORS LOCATED IN THE FIELD. THE CONSERVATION SERVICE WORKS WITH STATE AGENCIES IN PROVIDING DESCRIPTORS: SOIL CONSERVATION; PERSONNEL; SOIL MAPPING ; SOIL SOIL CLASSIFICATION; SOIL SURVEYS; SOIL SCIENCE

?